JAWS S3 98 Conference

Las Vegas, NV

15-18 June 98

Volume 1

Jaws S3 98 Conference

-AQ#U99-02-0191

PLEASE CHECK THE APPROPRIATE BLOCK BELOW:

□ ____________ copies are being forwarded. Indicate whether Statement A, B, C, D, E, F, or X applies.

☒ DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE: DISTRIBUTION IS UNLIMITED

□ DISTRIBUTION STATEMENT B:
DISTRIBUTION AUTHORIZED TO U.S. GOVERNMENT AGENCIES ONLY; (Indicate Reason and Date). OTHER REQUESTS FOR THIS DOCUMENT SHALL BE REFERRED TO (Indicate Controlling DoD Office).

□ DISTRIBUTION STATEMENT C:
DISTRIBUTION AUTHORIZED TO U.S. GOVERNMENT AGENCIES AND THEIR CONTRACTORS; (Indicate Reason and Date). OTHER REQUESTS FOR THIS DOCUMENT SHALL BE REFERRED TO (Indicate Controlling DoD Office).

□ DISTRIBUTION STATEMENT D:
DISTRIBUTION AUTHORIZED TO DoD AND U.S. DoD CONTRACTORS ONLY; (Indicate Reason and Date). OTHER REQUESTS SHALL BE REFERRED TO (Indicate Controlling DoD Office).

□ DISTRIBUTION STATEMENT E:
DISTRIBUTION AUTHORIZED TO DoD COMPONENTS ONLY; (Indicate Reason and Date). OTHER REQUESTS SHALL BE REFERRED TO (Indicate Controlling DoD Office).

□ DISTRIBUTION STATEMENT F:
FURTHER DISSEMINATION ONLY AS DIRECTED BY (Indicate Controlling DoD Office and Date) or HIGHER DoD AUTHORITY.

□ DISTRIBUTION STATEMENT X:
DISTRIBUTION AUTHORIZED TO U.S. GOVERNMENT AGENCIES AND PRIVATE INDIVIDUALS OR ENTERPRISES ELIGIBLE TO OBTAIN EXPORT-CONTROLLED TECHNICAL DATA IN ACCORDANCE WITH DoD DIRECTIVE 5230.25. WITHHOLDING OF UNCLASSIFIED TECHNICAL DATA FROM PUBLIC DISCLOSURE. 6 Nov 1984 (Indicate date of determination). CONTROLLING DoD OFFICE IS (Indicate Controlling DoD Office).

□ This document was previously forwarded to DTIC on ____________ (date) and the AD number is __________________.

□ In accordance with provisions of DoD instructions. the document requested is not supplied because:

□ It will be published at a later date. (Enter approximate date. if known).

□ Other. (Give Reason)

DoD Directive 5230.24, "Distribution Statements on Technical Documents," 18 Mar 87, contains seven distribution statements, as described briefly above. Technical Documents must be assigned distribution statements.

Tom Minnich
per phone call with J. Camp 10/22/98
Authorized Signature/Date

AFRL/FSD
2241 Avionic Circle
Wright -Patterson AFB, OH 45433-7318

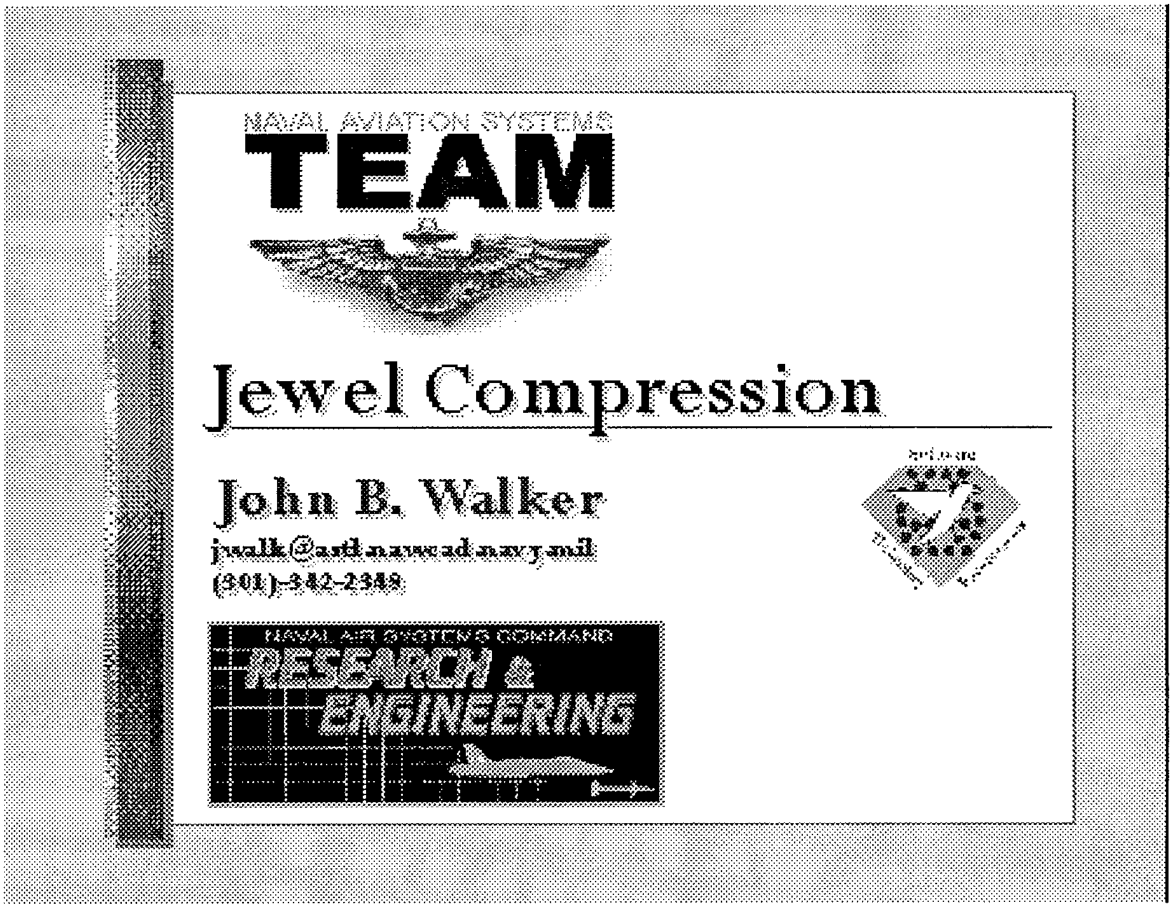
John Camp
Print or Type Name

937 255-2164 3562
Telephone Number

JAWS S3 98 Conference Volume 1

Table of Contents

Monday, 15 June 1998

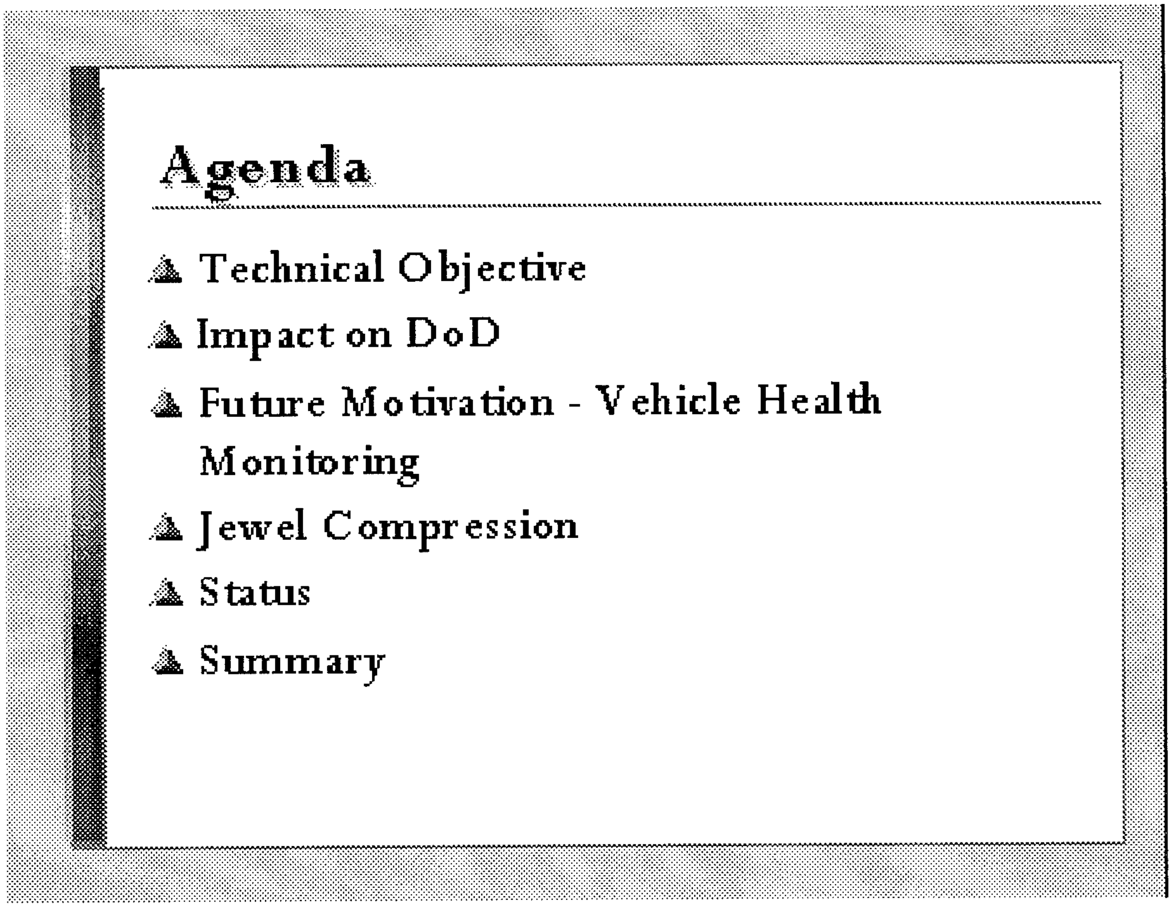


NAVAL AVIATION SYSTEMS
TEAM

Jewel Compression

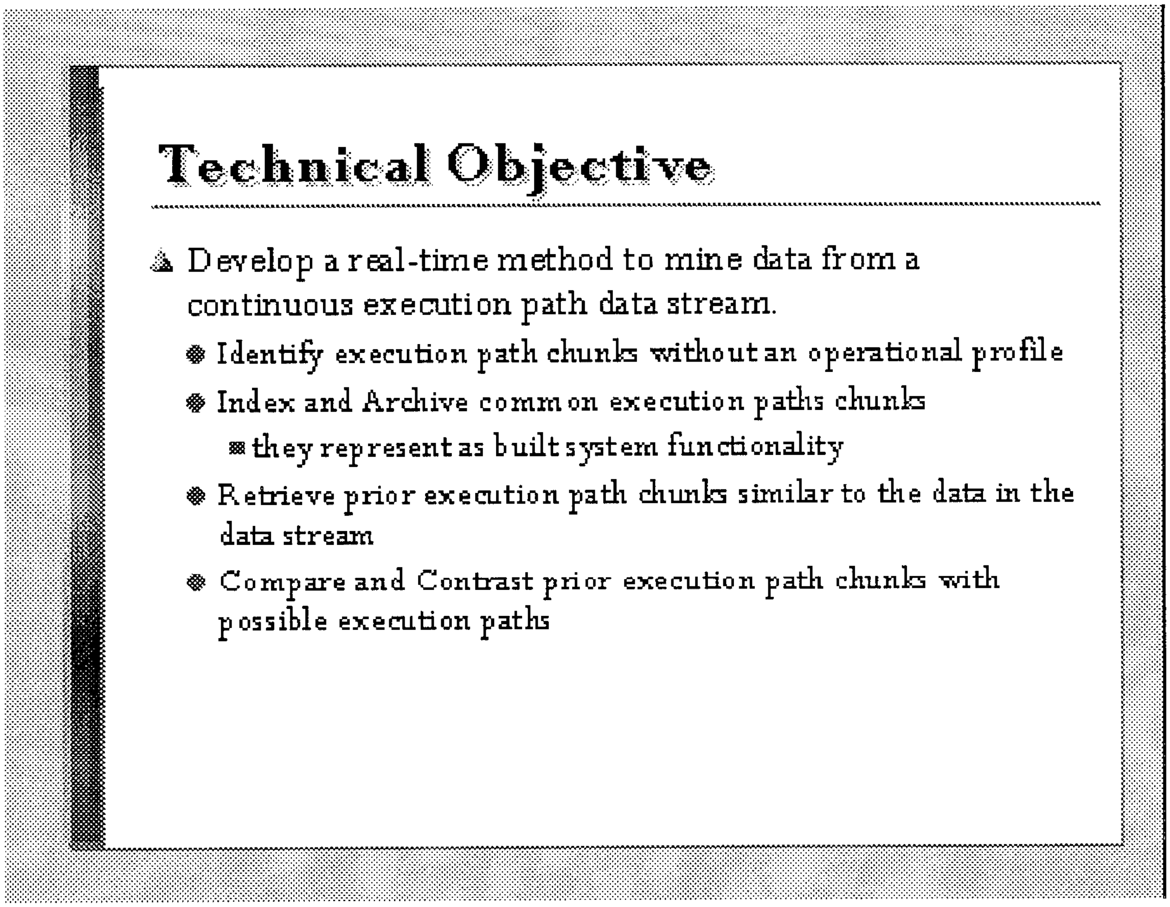
John B. Walker
jwalk@astl.nawcad.navy.mil
(301)-342-2348

NAVAL AIR SYSTEMS COMMAND
RESEARCH & ENGINEERING



Agenda

- Technical Objective
- Impact on DoD
- Future Motivation - Vehicle Health Monitoring
- Jewel Compression
- Status
- Summary

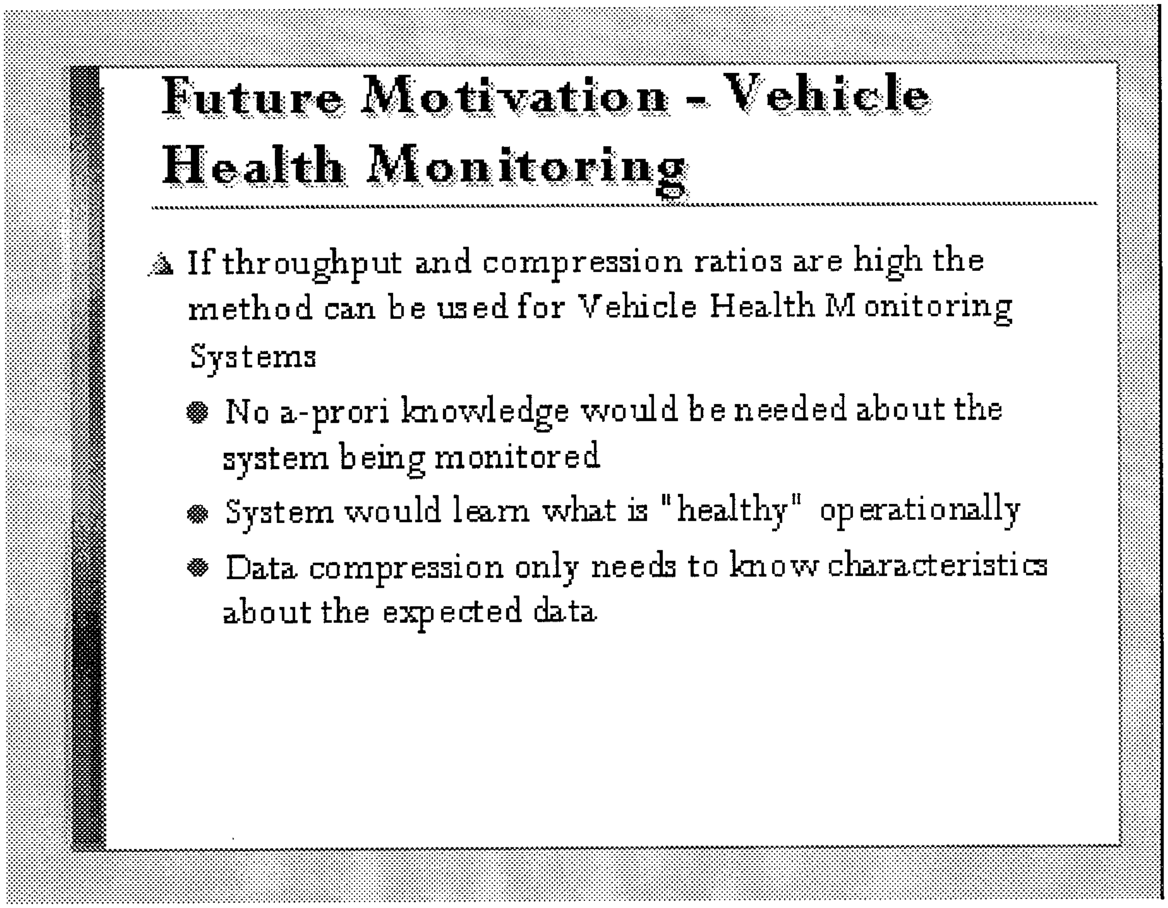


Technical Objective

- Develop a real-time method to mine data from a continuous execution path data stream.
  - Identify execution path chunks without an operational profile
  - Index and Archive common execution paths chunks
    - they represent as built system functionality
  - Retrieve prior execution path chunks similar to the data in the data stream
  - Compare and Contrast prior execution path chunks with possible execution paths

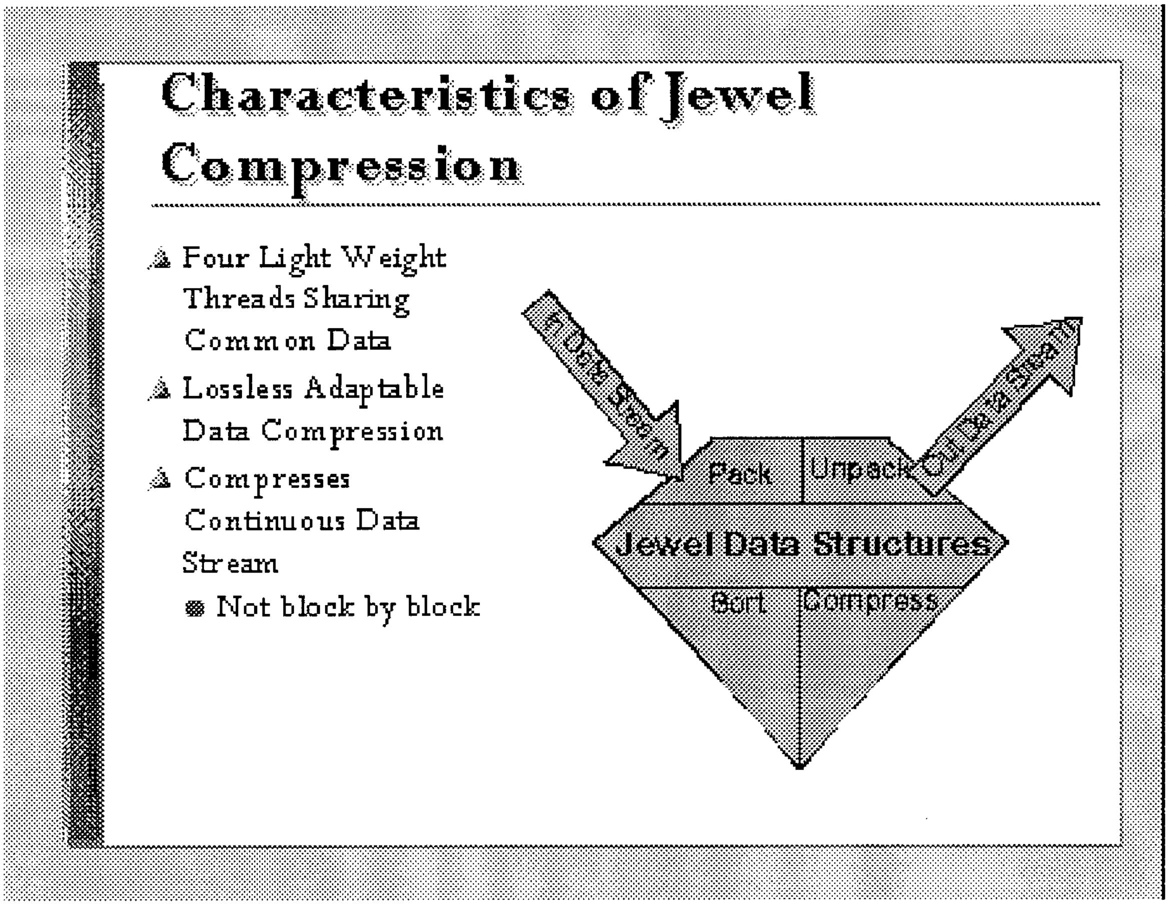
Impact on DoD

- Greater Return On Investment for Regression Testing
  - Regression testing can be done cheaper and faster for the same level of assurance
- Ability to Capture Knowledge From A Deployed System
  - Data packages require extensive maintenance and lag behind what is in use
  - Technical expertise is always being lost
  - Block upgrades and retrofits cause uncertainty in testing requirements
- Affordable Regression Testing Will Facilitate Affordable Readiness
  - A system is ready if its testing is current and complete
  - Regression testing that costs too much or takes too long will not be done
  - Testing with uncertain results is not complete



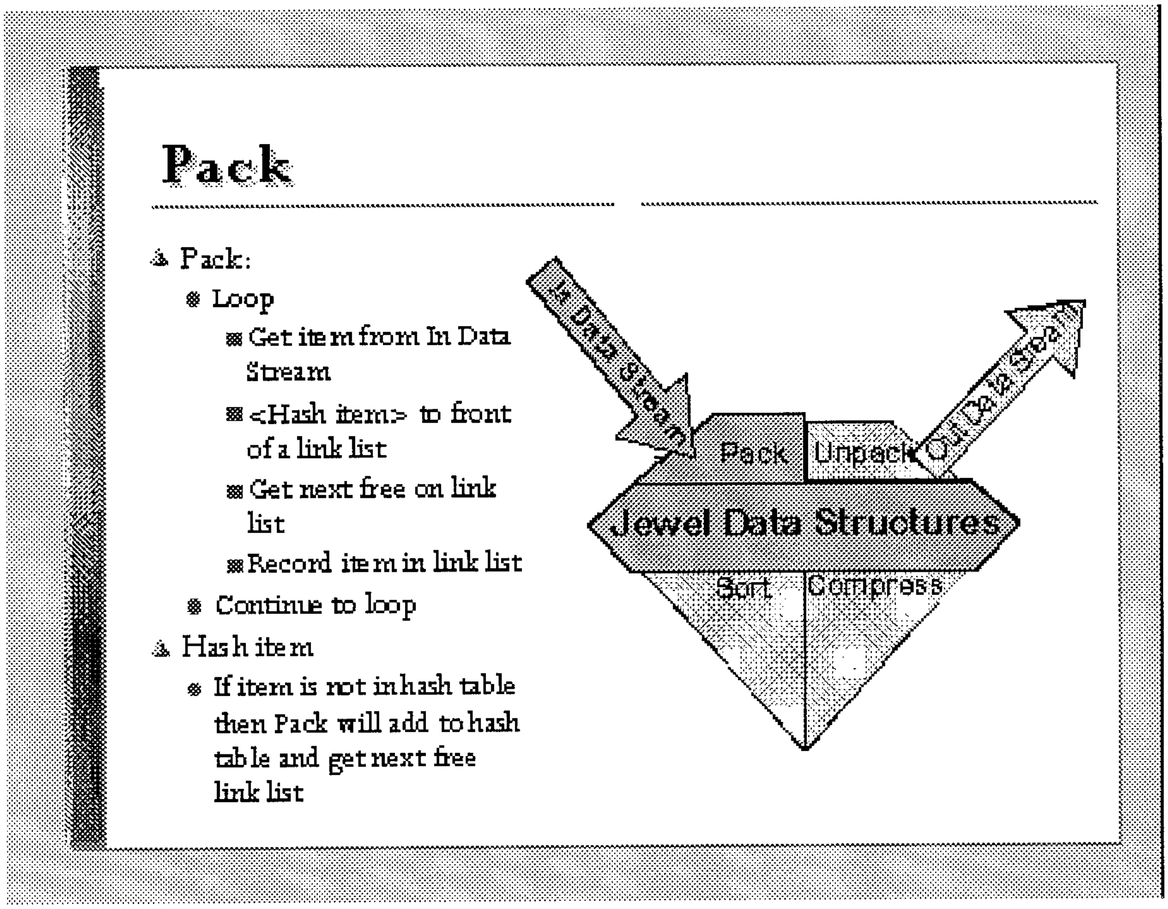
Future Motivation - Vehicle Health Monitoring

- If throughput and compression ratios are high the method can be used for Vehicle Health Monitoring Systems
  - No a-prori knowledge would be needed about the system being monitored
  - System would learn what is "healthy" operationally
  - Data compression only needs to know characteristics about the expected data



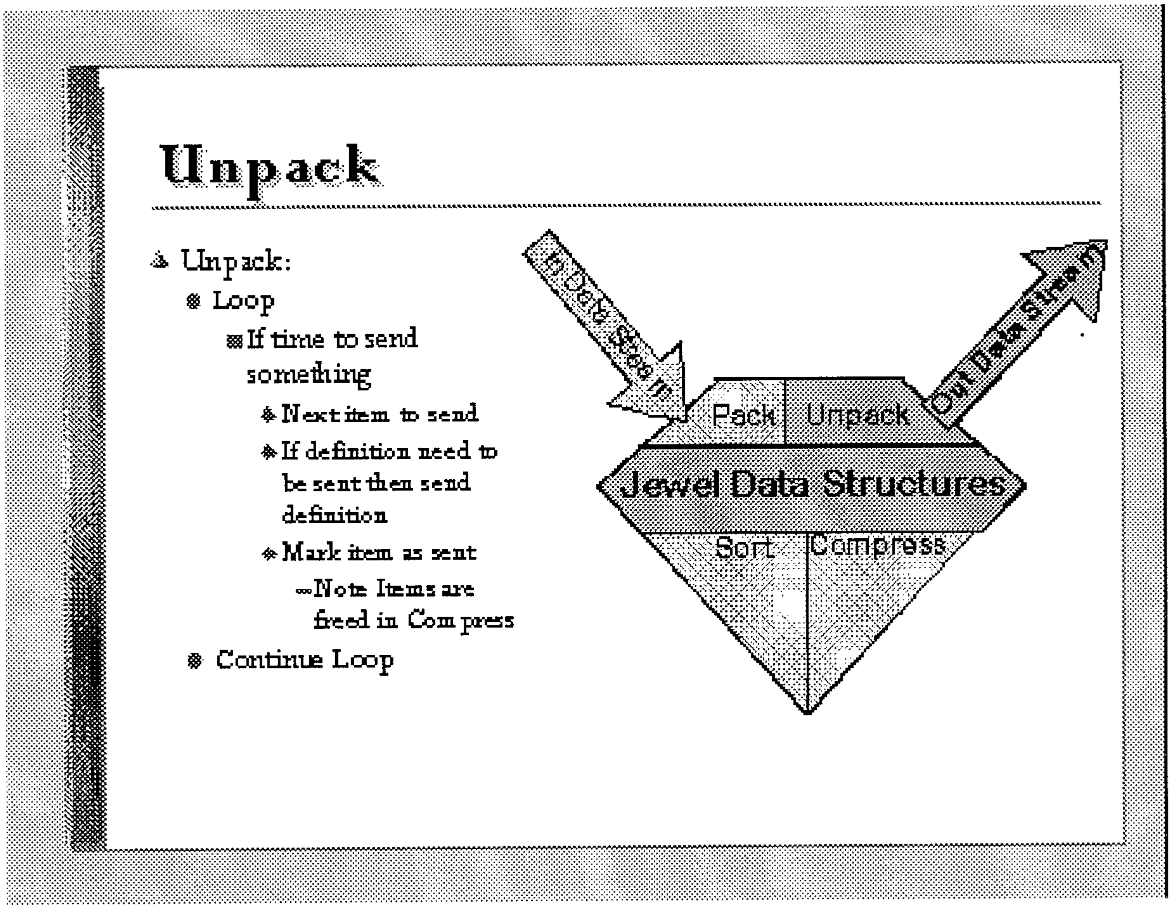
Characteristics of Jewel Compression

- Four Light Weight Threads Sharing Common Data
- Lossless Adaptable Data Compression
- Compresses Continuous Data Stream
  - Not block by block



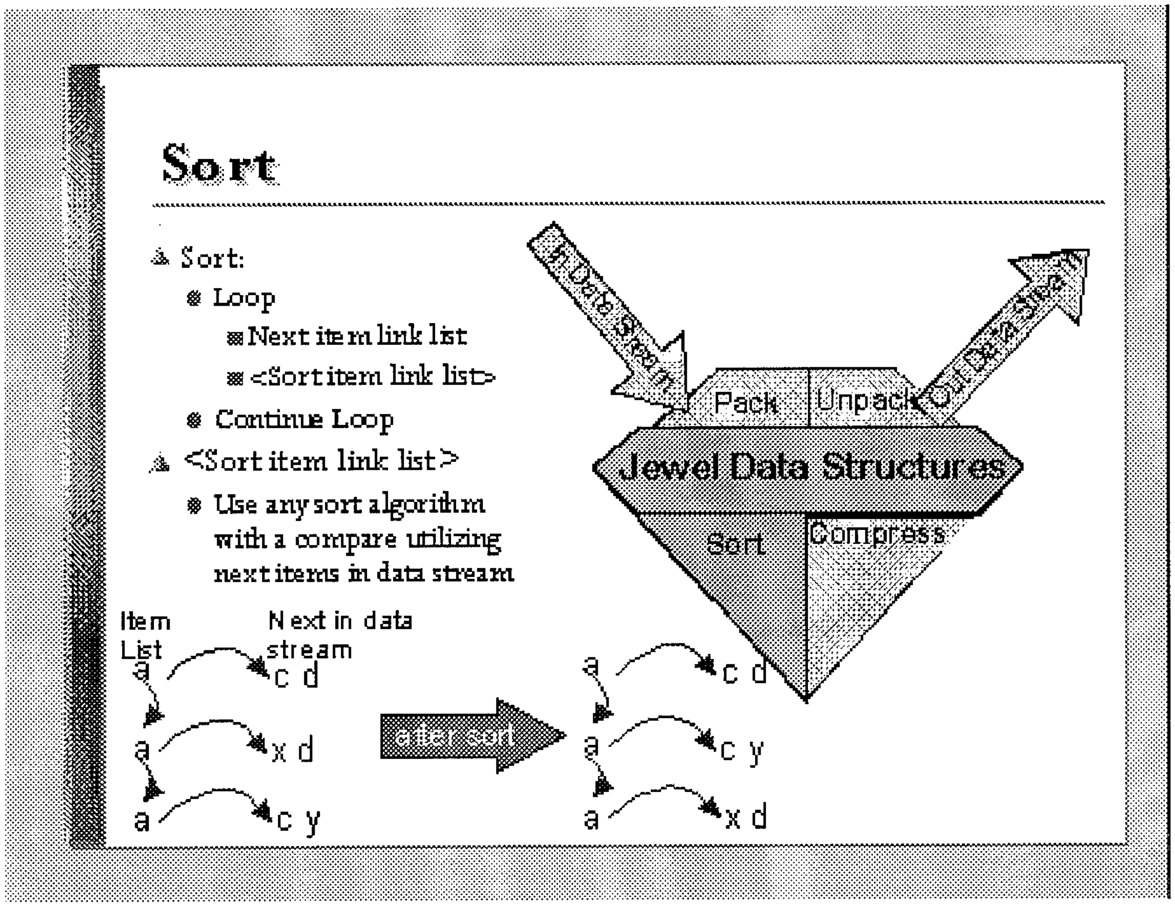
Pack

- Pack:
  - Loop
    - Get item from In Data Stream
    - <Hash item> to front of a link list
    - Get next free on link list
    - Record item in link list
  - Continue to loop
- Hash item
  - If item is not in hash table then Pack will add to hash table and get next free link list



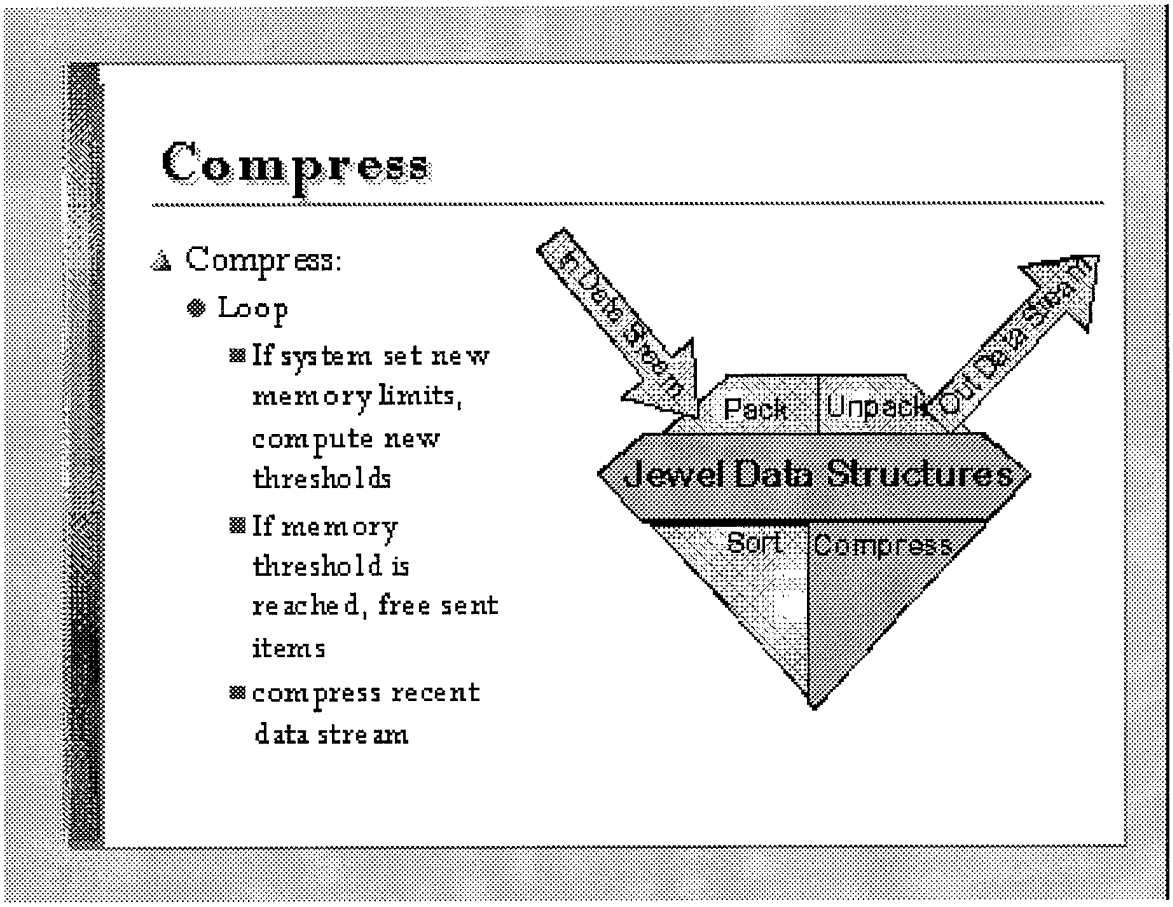
Unpack

- Unpack:
  - Loop
    - If time to send something
      - Next item to send
      - If definition need to be sent then send definition
      - Mark item as sent
        - Note Items are freed in Compress
  - Continue Loop



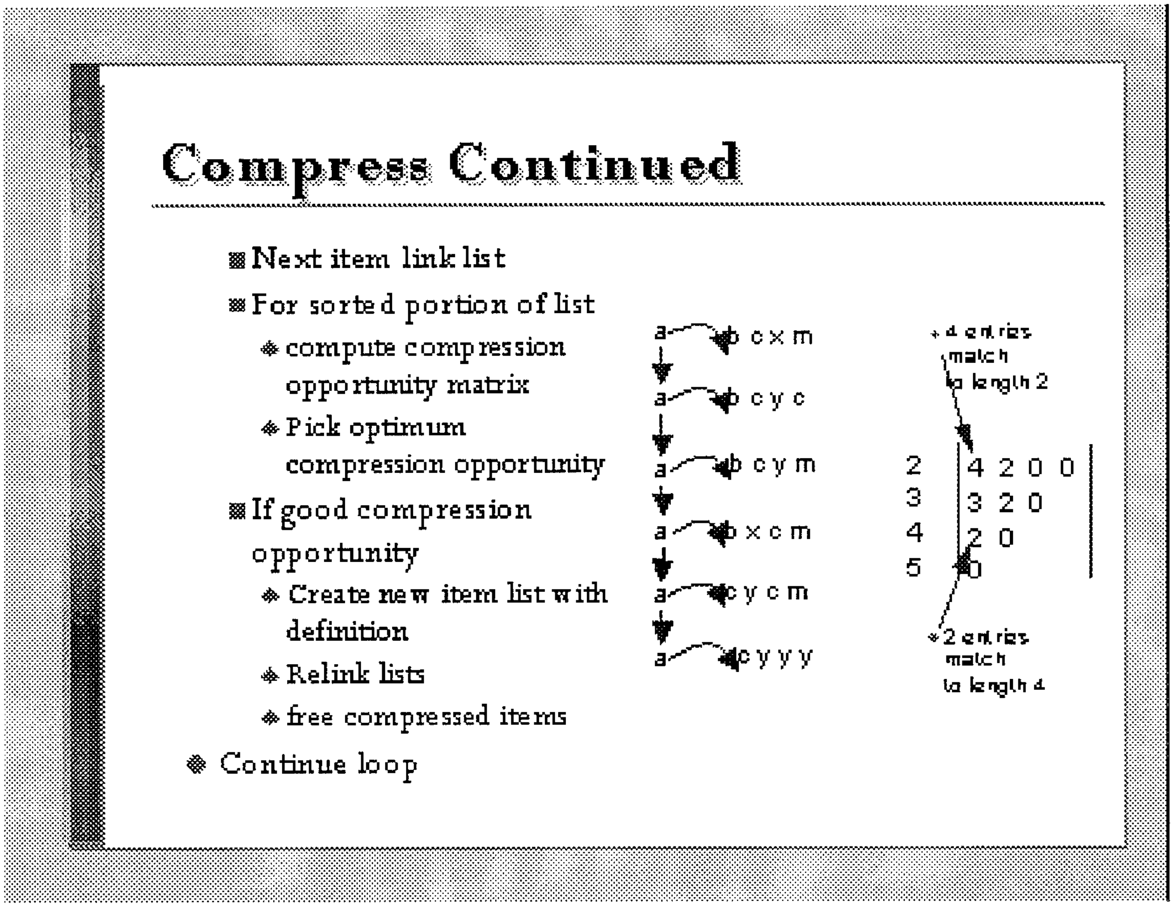
Sort

- Sort:
  - Loop
    - Next item link list
    - <Sort item link list>
  - Continue Loop
- <Sort item link list>
  - Use any sort algorithm with a compare utilizing next items in data stream



Compress

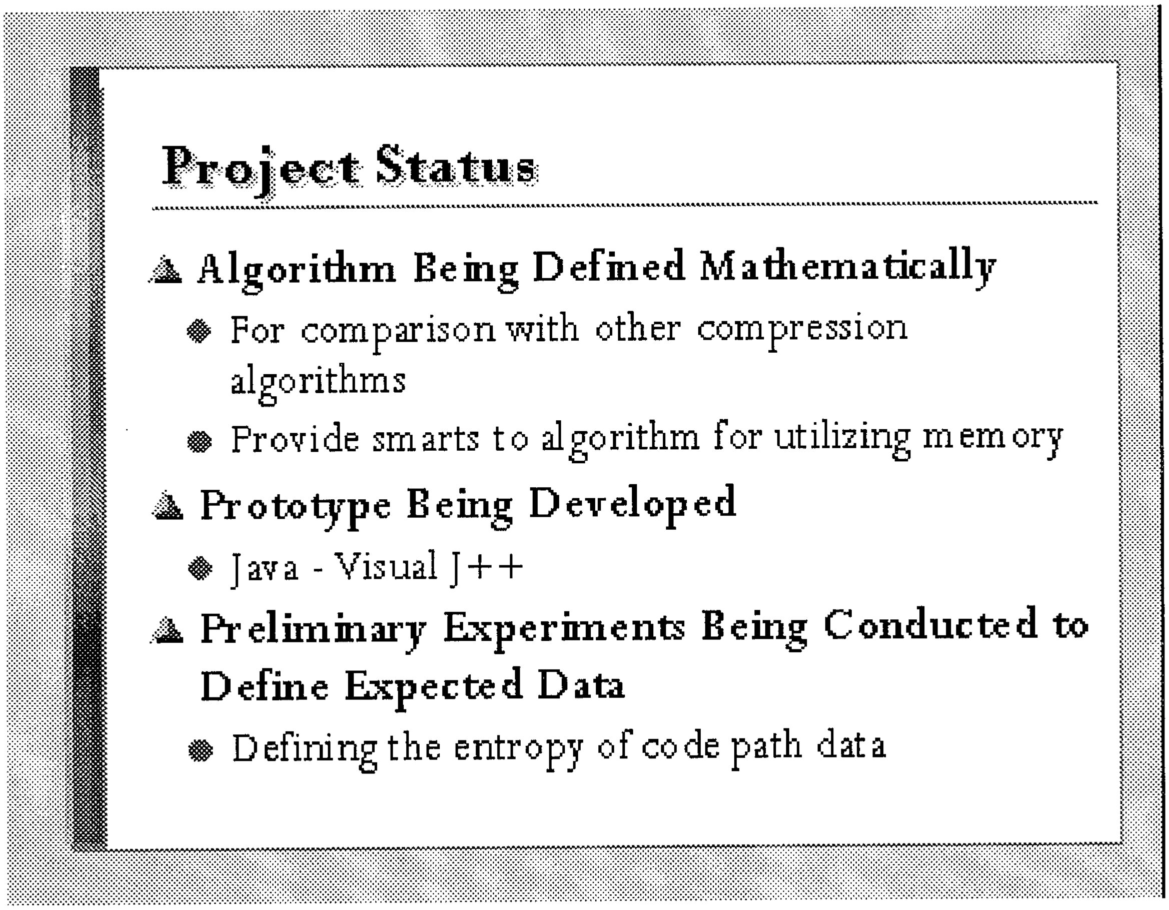
- Compress:
  - Loop
    - If system set new memory limits, compute new thresholds
    - If memory threshold is reached, free sent items
    - compress recent data stream



Compress Continued

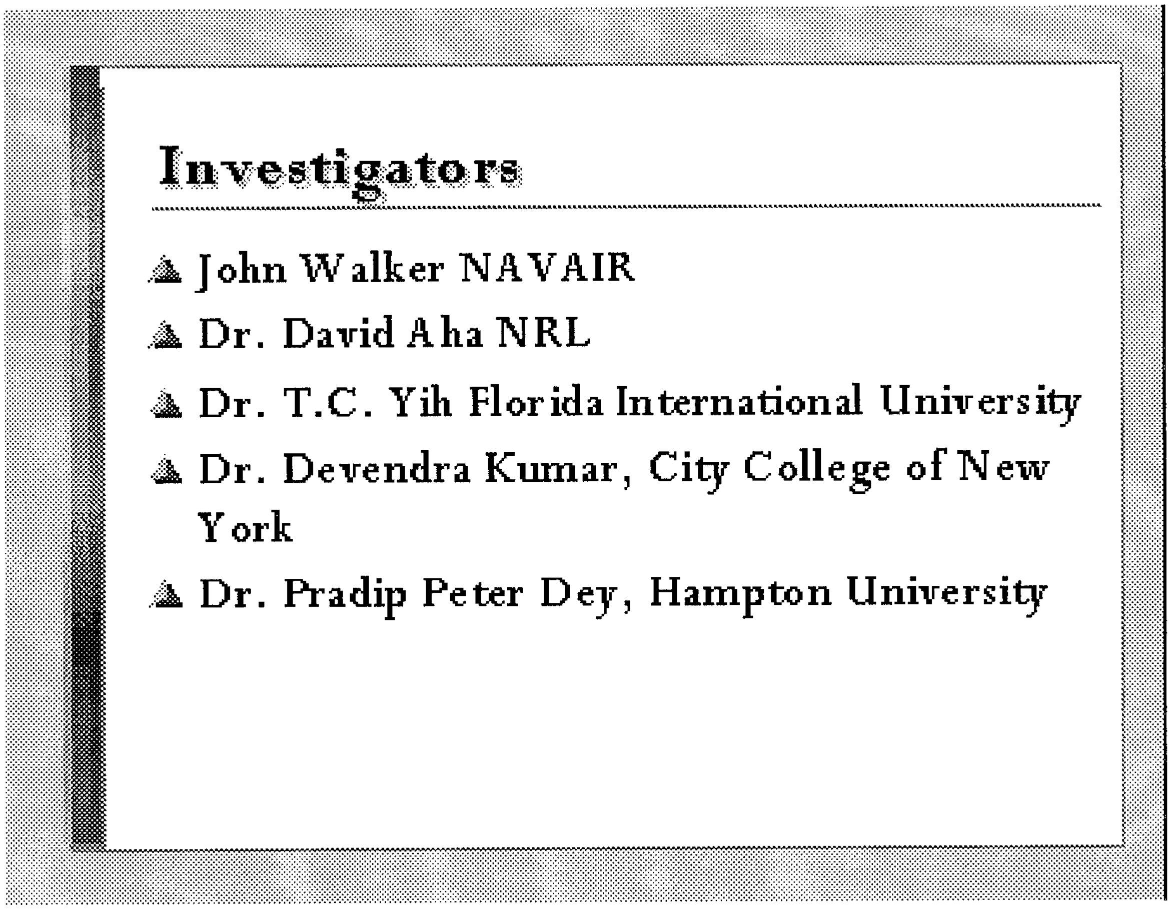
- Next item link list
- For sorted portion of list
  - compute compression opportunity matrix
  - Pick optimum compression opportunity
- If good compression opportunity
  - Create new item list with definition
  - Relink lists
  - free compressed items
- Continue loop

$$\begin{array}{c|cccc|}
2 & 4 & 2 & 0 & 0 \\
3 & 3 & 2 & 0 & \\
4 & 2 & 0 & & \\
5 & 0 & & &
\end{array}$$



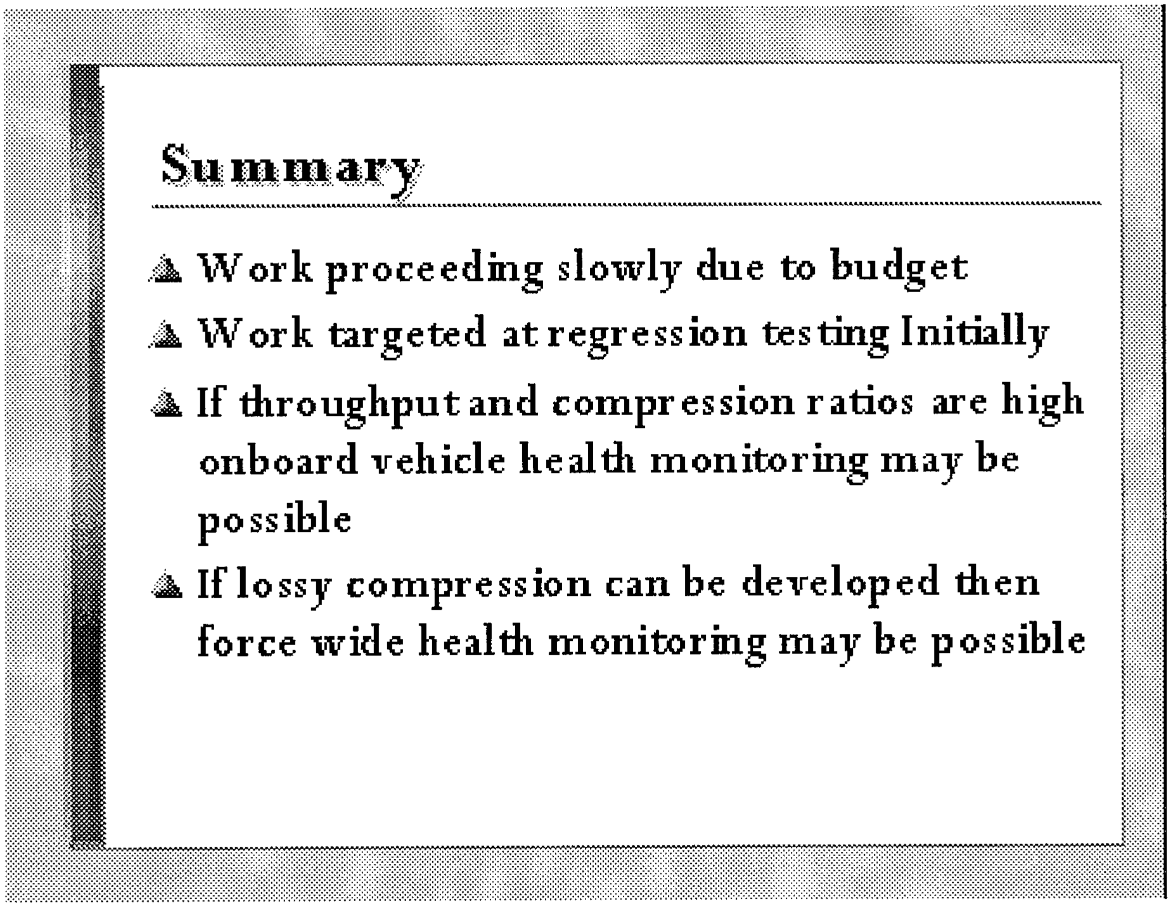
Project Status

- Algorithm Being Defined Mathematically
  - For comparison with other compression algorithms
  - Provide smarts to algorithm for utilizing memory
- Prototype Being Developed
  - Java - Visual J++
- Preliminary Experiments Being Conducted to Define Expected Data
  - Defining the entropy of code path data



Investigators

- ▲ John Walker NAVAIR
- ▲ Dr. David Aha NRL
- ▲ Dr. T.C. Yih Florida International University
- ▲ Dr. Devendra Kumar, City College of New York
- ▲ Dr. Pradip Peter Dey, Hampton University



Summary

- Work proceeding slowly due to budget
- Work targeted at regression testing Initially
- If throughput and compression ratios are high onboard vehicle health monitoring may be possible
- If lossy compression can be developed then force wide health monitoring may be possible

A Strong Partitioning Protocol for VME-based Systems Integration

Mohamed F. Younis **Jeffrey X. Zhou**

AlliedSignal Inc.
Advanced Systems Technology Group
9140 Old Annapolis Road
Columbia, MD 21045
Younis|Zhou@batc.allied.com

Abstract

This paper describes a new technology that can provide a global VME memory management in supporting the need of strong partitioning among multi-processor applications on the VME backplane. The technology allows each VME board to maintain a fault containment region on board level and prohibit fault propagation through the bus. The described techniques do not require any modification in the standard and the existing boards, and consequently, maintains the plug-and-play advantage of the VMEbus hardware products. The approach is to use a message passing mechanism for multiprocessing instead of shared memory. A software layer has been developed to support message-based inter-module communications using the available features provided by any standard VME card and still detect errors and prevent fault propagation.

1. Introduction

Advancements in technology have enabled the avionics industry to develop new design concept, which results in highly integrated software-controlled digital avionics. The new approach, referred to as Integrated Modular Avionics (IMA), introduces methods which can achieve high levels of reusability and cost effectiveness compared to earlier implementations of avionics [1]. The IMA approach encourages partitioning and using standardized building blocks in building environmental and functional components of avionics. Strong functional partitioning facilitates integration, validation and FAA certification. Following the IMA guidelines, the cost of both development and maintenance is expected to decrease because of large quantity production of the building blocks, lower levels of spares, and reduced certification costs.

The goal of this work is to develop a fault tolerant architecture that encourages functional partitioning and use of commercial-off-the-shelf (COTS) technology in implementing integrated control systems for both commercial and military aircraft. Integration of electronic controls is highly demanded by OEMs to reduce weight and maintenance costs. In addition, the integrated approach allows sharing of common resources such as power supplies, cooling system, data bus, etc. The use of a fault tolerant architecture for the integrated control is a necessity to minimize the risk of loosing multiple control functions due to some hardware or software failure in the unified environment.

The airborne backplane bus is one of the most important components in integrated modular avionics. While much backplane bus designs have been proposed, only a few are actually used. Selecting the backplane bus is affected by many design and engineering factors, such as performance, reliability, and fault-tolerance. Although, such issues are very important to ensure certain level of safety of commercial jet aircraft and high availability of military aircraft, the cost of the bus and associated line replaceable modules is a major concern.

Most of the currently available dependable backplane bus systems are expensive and supplied by very few vendors, such as ARINC 659[2]. It is clear that there is a need for an affordable bus system that provides the required levels of dependability for an integrated utility control and complies with the IMA design methodology. This paper shows how to enhance functionality and overcome inefficiencies in commonly used and widely manufactured low-cost buses to make them suitable for an integrated control. Since the trend in implementing today's real-time embedded applications is toward the use of commercial-off-the-shelf open architecture to reduce costs and facilitate system integration, the VMEbus system [3] is a prime candidate because it is both rigorously defined and widely supported. In addition, there is an expanding selection of VMEbus boards and vendors that guarantee competitive prices and continuous support. Moreover, the VMEbus offers an open architecture that facilitates the integration of multiple vendors' boards. Such features make the VMEbus an attractive choice for integrated utility control.

However, the VMEbus standard does not impose strong functional partitioning and allows fault propagation from one board to another, as we discuss in *Section* 3. Such weakness limits the use of the VMEbus in an integrated control system. This paper presents techniques for strong partitioning of multi-processor applications that maintain fault containment on the VMEbus. The suggested techniques do not require any modification in the standard and the existing boards, and consequently maintain the cost advantage of the VMEbus hardware products. In addition, a fault tolerant architecture is proposed for an integrated control system. The architecture includes commercial-off-the-shelf hardware and software components and other AlliedSignal- developed products to provide a reliable and cost-effective integrated control platform.

2. An Overview of the VMEbus

The VMEbus design is highly influenced by the development of the Motorola MC68000 microprocessor, although nowadays VME boards are available for many other processors. The VMEbus allows multi-processing, expandability and adaptability by many design and processors. It handles data transfer rates at speeds in excess of 40 Mbytes/sec using parallel data transfer.

The VMEbus is asynchronous and non-multiplexed. Because it is asynchronous no clocks are used to coordinate data transfer. The system clock is 16Mhz and has no relation to the other bus activities. Typical uses of it include bus timers, memory refresh circuits, serial I/O time bases and synchronous state machines. Data is passed between modules using interlocked handshaking signals where cycle speed is set by the lowest module participating in the cycle. Using asynchronous protocol, in the VMEbus, provides reasonable compatibility to integrate products from various vendors.

Master-slave architecture is used in the VMEbus. Modules can be designed to act as master, slave or both, although some modules can be configurable. Before a master can transfer data it must first acquire the bus using a central arbiter. This arbiter is part of a module called the system controller. Its function is to determine which master gets the next access to the bus. The bus arbiter grants the bus according to prioritized requests handling or simply using round robin. Bus grant signals are propagated back to the requester through a daisy chain over all modules between the requester and the system controller. Four levels of arbitration are provided, each with its own daisy chain. Every module has a set of jumpers to select the arbitration level. The bus arbiter may prioritize requests according to its arbitration level.

All interrupt service routines are user defined. The daisy chain again is used to propagate the acknowledgement signal from module to module until it reaches the interrupter. The interrupter then places a STATUS/ID on the bus to notify the handler of which card initiates the interrupt.

Due to the daisy chaining used for bus mastership requests and interrupt, modules are position-sensitive. The first slot is reserved for the system controller. The closer the module to the system controller, the higher the privilege of getting the data bus and the higher the priority of this modules interrupts.

The VMEbus provides support for multiprocessing using shared memory. To avoid inconsistency while updating shared memory, read-modify-write bus cycles are to be used. The read-modify-write cycle allows updating shared memory as an atomic transaction and prevents race conditions.

Although the VMEbus does provide reasonable compatibility to integrate products from various vendors, fast parallel data transfer, and a wide support by many manufactures, fault-tolerance in VMEbus based systems is very limited. The next section discusses fault detection mechanisms in the VMEbus and elaborates weaknesses in fault containment and fault-tolerance.

3. Fault-Tolerance Challenges in the VMEbus

The VMEbus relies on modules for detecting and reporting faults on a specific failure control line. VMEbus modules are expected to have on-board firmware diagnostics to detect faults. In response, the system controller polls every module by reading its status register to identify the faulty module. Generally, the build-in-test and transmission time-out provides limited fault coverage for only permanent faults.

The VMEbus master (sender) monitors the time for data transfer. If the receiver does not acknowledge the message, the master times out data transfer and retransmit. The bus does not provide error detection or correction for the transferred data. There is no redundancy in either the transmission lines or the transferred data on the bus. In addition, the VMEbus system allows a single point of failure by using a centralized mastership control of the bus. The system controller organizes all communications between modules. Moreover, the system controller is responsible for monitoring the failure control line and for tracing the faulty module, which derived the failure line. Faults in the system controller can bring down the whole system.

The shared memory model used by the VMEbus for multiprocessing makes the modules tightly coupled. In the absence of message verification, faults can propagate from one module to the others. Errors cannot be contained within the faulty module and can jeopardize the behavior of the whole system.

The daisy chain reliability is questionable. Each module either accepts the bus grant or passes it over to the next module. Failure of one module may break the chain and affect other modules. Breaking the chain can prevent other modules from getting mastership of the bus and from reacting to interrupts. Communication between modules can be highly affected and the whole system can break down.

From the former discussion we can conclude that the VMEbus needs enhancements to strengthen its fault-tolerance capabilities, specifically in containing errors and recovery from failure. The following issues need to be addressed in order to improve containment of faults in a VMEbus system:

1. validating the inter-module data transfer.
2. propagating faults through the use of shared memory.
3. breaking the daisy chain because of a module failure.

In addition, fault recovery mechanisms need to be added to:

1. prevent having the system controller as a single point of failure.
2. allow redundancy for the system critical functions.

The next sections provide a discussion of our approach to address these issues starting with techniques for fault containment.

4. A Fault tolerant VMEbus

Because low cost is an important feature of the VMEbus, enhancing the fault containment capabilities should avoid changing the design and the layout of the currently available cards. Changing the design a VME card will not only require reengineering and revalidation which increases the manufacturing cost, but also will again limit the number of vendors who agrees to do the modifications. Thus, the suggested approach should be constrained by preserving the current hardware design of the cards as much as possible.

As illustrated in the previous section, both fault containment and failure recovery features need to be improved. In the following subsections, we present our approach to enhance these features.

4.1 Fault Containment Techniques

As illustrated in the previous section, fault containment on the VMEbus needs to be added. In the following subsections, our approach is discussed.

4.1.1 Inter-module Data Transfer

The VMEbus features parallel data transfer between modules. There are no error detection or correction bits associated with the transmitted data. Adding such bits will significantly affect the VME card design and, therefore, is not an option. As an alternate approach, an error detection code, e.g. cyclic redundancy check, can be appended to the end of the data. The message transmission module within the operating system kernel can generate the code. Although the software-generated error detection code is less efficient than the hardware-based implementation, no card redesign is necessary using the software approach. For higher dependability, an error correction code can be appended. Because that error detection/correction code will reduce the efficiency of the data transfer on the bus and consequently the performance, it may be possible through the kernel to dynamically select either to append error detection or error correction code according to the length of the transmitted data. The receiver module should validate the data using the error detection/correction code before committing that received data.

Using such information redundancy within the transferred data fits the multiprocessing scheme proposed in the next subsection.

4.1.2 Multiprocessing

Strong partitioning of modules is one the most important IMA requirements which the VMEbus lacks. Multiprocessing in the VMEbus uses a shared memory mechanism that allows faults in one module to cause errors in other non-faulty modules by writing to their memories. In our approach, we used a message-passing mechanism instead. The challenge is to support message-based inter-module communications using the available features provided by the VME cards and still detect errors and prevent fault propagation.

To support messages, a buffer is to be declared and dedicated only for messages. A message buffer is the only globally visible memory of a module. Modules are not allowed to access the memory of other modules other than their message buffers. In addition, access to a message buffer is restricted to read-only for modules that do not own that buffer. No card is supposed to write to the memory of other cards. If a master wants to send data to a slave, it simply writes a message for the slave into the master's message buffer. The slave reads that message from the master memory and reacts to it.

A specific message format can be imposed that contains the sender ID, receiver ID, error detection or correction code, and a message unique ID. The sender should perform error detection and correction encoding. The slave will check the contents of the message before reacting to it. The receiver can detect addressing errors in the message by verifying the sender ID and receiver ID. In addition, transmission errors can be detected or recovered using the information redundancy in the form of the error detection or correction code in the message.

Synchronization can be achieved either by polling the message buffer of the sender for the required message, or by using the address monitoring feature provided by the VMEbus to interrupt the receiver as soon as a message is being written by the sender in the designated address. The message ID can be useful to overcome race conditions if the receiver tried to read the message before it is ready which may be is possible if the VMEbus has a higher priority than the local bus. The message buffer can be partitioned for various cards and slaves can expect a unique location for their messages. The adopted application execution-synchronization mechanism is a designer decision.

Using this technique, errors in the sender can be isolated and prevented from propagation to the receiver. A fault in the sender may affect the receiver only through the message. As explained earlier, no write permission will be granted for a card to the memory of others. Errors in the message can be either in data, sender ID, receiver ID, Message ID, or message format. The receiver should be able to detect errors in the message body by validating the message format, error detection code, sender ID and the receiver ID. The message ID can be checked to guarantee the right message sequence. Any error in the message detected by the receiver will invalidate the entire message and a recovery action will be taken. An addressing fault in the receiver that may get it to read from the wrong card or the wrong address within the right card will affect the message format and the sender ID. Furthermore, the mapping of message buffers of cards in the global address space of the VME system should be widely distributed so that an addressing error can not change a valid global address into another valid address. Maintaining a suitable hamming distance can guard the system against permanent or transient stuck failure of one or more address bits. Thus, the system will be functionally partitioned. Faults can be within the faulty module and will not affect other modules.

4.2 VMEbus failure

The VMEbus relies on a centralized control for arbitrating the bus mastership among various cards connected to the bus. The centralized control is vulnerable to a single point of failure that can bring down the whole system. In addition, a daisy chain is used in the VME backplane to propagate bus grant and interrupt acknowledgment signal. A failure in one module may break the chain and affect the following modules in the chain preventing them from communicating over the VMEbus.

One approach to tolerate failure of the VMEbus is to provide an alternate path for inter-module communication as shown in *Figure 1.* Many vendors include a secondary bus on their VME boards such as the *VME Subsystem Bus* (VSB) [5] and the *RACEway Interlink* [6] to act as a local subsystem extension bus. Both the VSB and the RACEway interlink allow the developer to reduce the VMEbus traffic by using the secondary bus for some DMA and I/O operations. Using either the VSB or the RACEway interlink to tolerate failures in VMEbus requires no modifications in the VME boards and allows the use of cost-effective off-the-shelf boards to develop avionics and other mission critical applications complying with the IMA specifications.

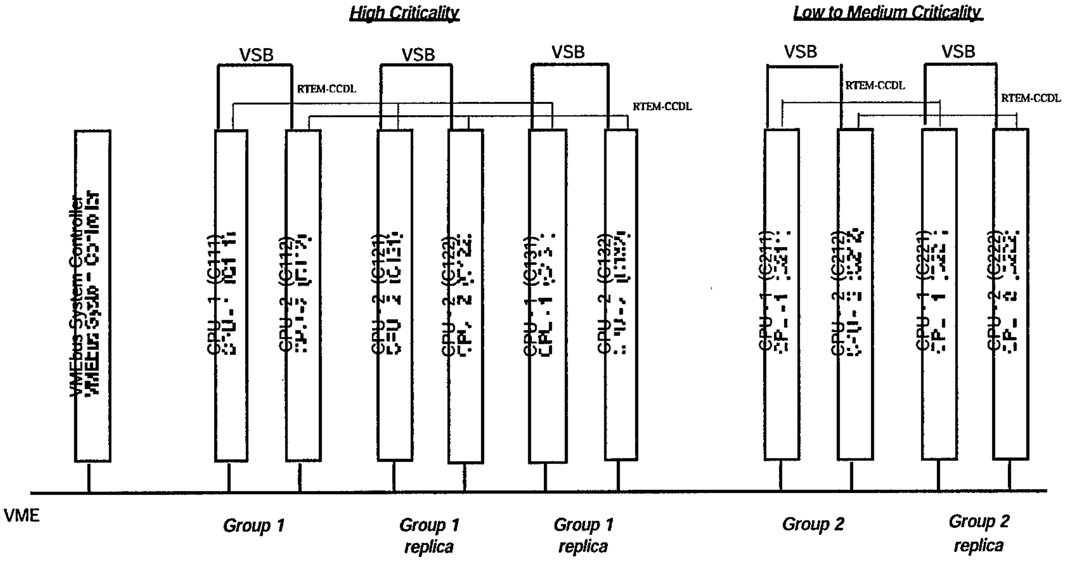


Figure 1: A Fault tolerant VME-based architecture using VSB secondary buses

Another approach for tolerance of VMEbus failure is to use separate redundant VME backplanes that fail independently, as shown in *Figure 2.* The redundant backplanes host modules capable of performing the same functions on the primary backplane.

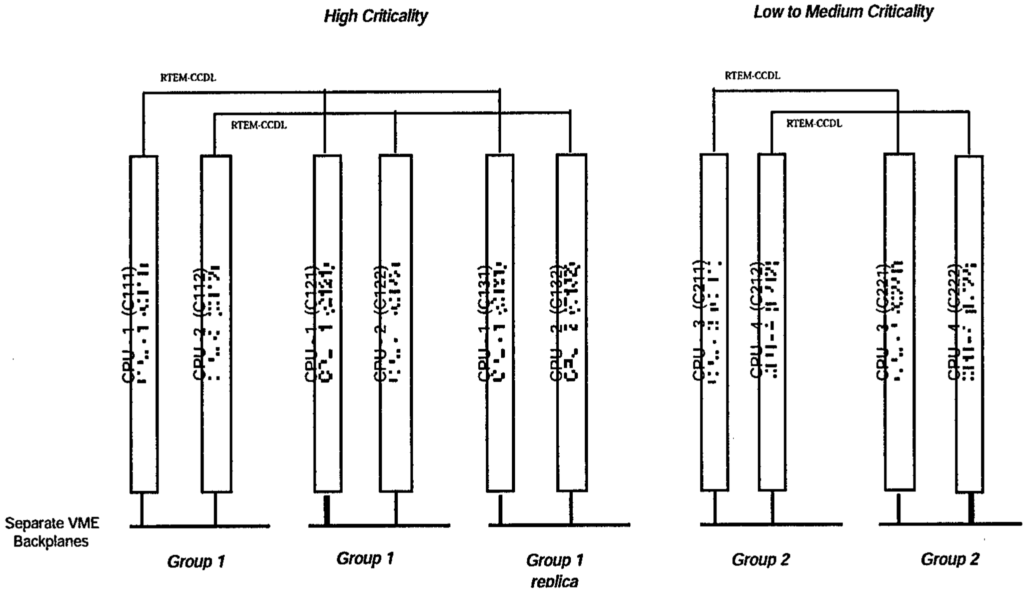


Figure 2: A Fault tolerant VME-based architecture using Separate VME backplanes

Although using a secondary bus or a redundant VME backplane can tolerate a VMEbus failure, it is necessary to allow replication of critical system components and manage such redundancy. In the next subsection, we discuss an approach to managing redundancy. We propose the use of Redundancy Management System (RMS) [7], a fault tolerant executive developed by AlliedSignal, for detecting and masking errors and for redundancy management.

4.3 Redundancy Management

The use of redundancy is very common in fault tolerant systems. Spare units are usually included that can be either passive or active during normal operation. Redundancy management is required to coordinate the interaction between the primary and spare units. Managing redundancy typically needs to be considered during the design and the development of the application.

One or more backup unit is used in active redundancy. A backup will be activated in case of primary unit failure. The backup needs to be updated frequently with changes in the primary internal state (processor execution state and program store). When activated, the backup will resume execution from the last update (checkpoint). The more the frequency of sending of checkpoints to the backup, the shorter the time needed for recovery. However, checkpointing increases the load on the primary unit and may affect its performance during normal operation. Active replication on the VMEbus does not require multiple backplanes. The application software may need careful design to accommodate checkpointing without dramatic effects on the performance. In addition, checkpoints needs to be safely scheduled so that it would not introduce contention on the bus and delay other essential traffic. Another challenge in active replication is related to fault coverage. The activation of a backup depends heavily of the detection of an erroneous condition within the primary processor. The fault coverage within the boards needs to be comprehensive which is hard to guarantee while using commercial-of-the-shelf boards.

Using passive redundancy, the spare units perform the same function performed by the primary unit. The output of all units is subject to voting to tolerate erroneous output from some faulty units. One way of applying this approach to the VMEbus requires the use of multiple backplanes, as the case in the vehicle management computer of the X-33 (experimental prototype for the single step to orbit space shuttle). Alternatively, voting can be performed internally within the same backplane using multiple modules, as shown in *Figure* 2. We intend to use the RMS for redundancy management. RMS ensures a synchronized and consistent execution for a module and its replicas. In addition, it is possible to support different levels of criticality by the number of replicas. For example, using triple redundancy, it is possible to mask permanent, intermitted and transient faults by RMS. For lower criticality functions, a dual redundancy can be used while RMS will provide error checking and will allow fail save operation. In addition, RMS will provide a homogenous error detection capability and fault coverage for all boards in the system regardless their vendor.

5. Fault-Tolerant Architecture

We propose the fault-tolerant architecture shown in *Figure* 2 for an integrated control system. Modules that need to communicate with each other are grouped together, `Group1` and `Group2` in *Figure* 2. Groups are replicated by providing similar modules running the same programs. RMS is used for redundancy management and error detection for the replicated groups. RMS consists of two major components; the Fault- Tolerance Executive (FTE) implemented in software, and the Cross Channel Data Link (CCDL) implemented as a daughter (mezzanine) board.

The VMEbus standard raises important issues in the location of replicated modules. As mentioned in *Section* 2, the priority of bus mastership and interrupt will be granted to the left-most module when multiple requests are made on the same arbitration or interrupt request line. Thus, a replica of a module needs to be assigned the same arbitration and interrupt request line of that module. This will allow predicting the bus schedule. In addition, groups of the highest criticality needs to be inserted in the left-most slots of the VME backplane, so as not to be affected by a failure of a low criticality group which breaks the daisy chain of the VMEbus. The message-passing inter-processor communication mechanism, presented in *Section* 4, will guarantee fault containment for modules within the same group using the VME bus.

The use of RMS in this architecture makes it possible to support different levels of criticality within the same integrated platform. Using dual redundancy with RTEM allows a fail-safe mode of operation. Increasing the number of replicas to three allows the fault masking, while the use of four replicas enable the handling of Byzantine disagreement between replicas. On the other hand, the use of independent backplanes, it is possible to tolerate a VMEbus bus failure. It should be noted that the fault-tolerance issues related to the power supply of the VME backplane is orthogonal and is beyond the scope of this project.

6. Summary

A proof-of-concept prototype has been built using COTS components as shown in *Figure 3.* The prototype includes three VME backplanes, each of them hosts two PowerPC processor boards. VxWorks, the real-time operating system from WindRiver Systems, is purchased and integrated with the hardware. The strong partitioning inter-processor communication protocol has been implemented and integrated within VxWorks. Initial testing has been performed and currently fault injection experiments are being conducted. In addition, the performance of multi-processing communication using the protocol is being measured. A software design document and a user manual have been prepared for application desingers.

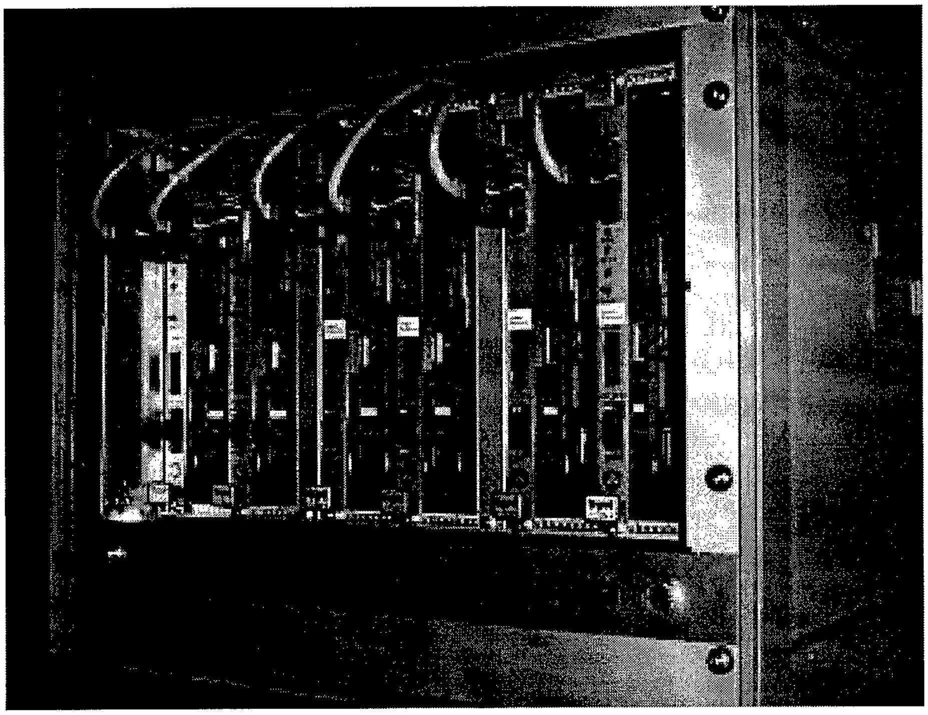


Figure 3: A proof-of-concept demonstration prototype

References

[1] ``Design Guide for Integrated Modular Avionics'', ARINC report 651, Published by Aeronautical Radio Inc., Annapolis, MD, November 1991.

[2] ``Backplane Data Bus'', ARINC Specification 659, Published by Aeronautical Radio Inc., Annapolis, MD, December 1993.

[3] ``IEEE Standard for a Versatile Backplane Bus: VMEbus'', std 1014-1987, Published by The Institute of Electrical and Electronics Engineers, New York, NY, March 1988.

[4] ``Motorola MVME162LX Embedded Controller Programmer's Reference Guide'', Motorola, Inc. Computer Group, Tempe, Arizona.

[5] ``IEEE Standard for a Multiplexed High-performance Bus Structure: VSB (ANSI) Based on IEC 821 Bus, IEC 821-1987'', std 1096-1988, Published by The Institute of Electrical and Electronics Engineers, New York, NY, 1988.

[6] ``RACEWAY Interlink-Data Link and Physical Layers'', ANSI/VITA 5-1994, RACEway Interlink, Published by The VMEbus International Trade Association (VITA), Scottsdale, AZ, 1994.

[7] Jeff Zhou, ``RTEM: A Generic Fault Tolerant Operating System for Real-Time and Mission Critical Applications'', Project Report, AlliedSignal Microelectronics and Technology Center, Columbia, MD, 1992.

Redundancy Management System for Fault-Tolerant Computing

Jeffrey X. Zhou Thomas G. Roden III
James Ernst Dar-Tzen Peng
Louis Bolduc Mohamed Younis

November 17, 1997

1. INTRODUCTION

Fault-tolerant computing is an important technology that assures correct computing results in the existence of faults and errors in a system. The use of redundancy is the primary method for fault tolerance. There are many different ways of managing redundancy in hardware, software, information and time. Due to various algorithms and implementation approaches, most current systems use proprietary design for redundancy management and the design is usually interwoven with application software and hardware.

The Redundancy Management System (RMS) described in this invention disclosure takes a different approach. The approach is to separate, physically or logically, the RMS from application development so that the overall design complexity of a system can be reduced. Using this approach, a system developer can design applications independently and rely on the RMS to provide redundancy management functions such as cross channel data communication, system synchronization, voting, error detection and recovery. RMS and application integration is accomplished by a programmable bus interface protocol which connects the RMS to application processors.

The RMS is a redundancy management methodology captured in a set of algorithms, data structures, operation processes and designs. It can be implemented in hardware, software, or hybrid. The RMS works with a distributed system which has redundant computing resources to handle component failures. The distributed system can have two to eight channels (or nodes) depending upon system reliability and fault tolerance requirements. A channel consists of a RMS and application processor(s). Channels are interconnected together through RMS's Cross-Channel Data Link (CCDL) module to form a redundant system. Since individual applications within a channel do not have full knowledge of other channel's activities, it is the RMS's responsibility to provide system synchronization, maintain data consistency, and form system-wise consensus of faults and errors occurred in various locations in the system. The technical challenge for the RMS design is to accurately establish and maintain a global agreement regarding the output data and the status of the system health even with the existence of maliciously [1] failed components in the system.

2. RMS SYSTEM MODEL

2.1 RMS Functions

RMS provides the following redundancy management functions:

Cross-channel data communication
Frame-based system synchronization

Data voting
Fault and error detection, isolation and recovery
Graceful degradation and self healing

The cross-channel data communication function is provided by the CCDL module. The CCDL module has one transmitter and has up to eight parallel receivers. It takes data from its local channel and broadcasts data to all channels including its own. Communication data is packaged into certain message formats and parity bits are used to detect transmission errors. All CCDL receivers use electrical to optical conversion in order to preserve electrical isolation among channels. Therefore, no single receiver failure can over drain current from other channel's receivers resulting in a common mode failure across the system.

The RMS is a frame-based synchronization system [1]. Each RMS has its own clock and the system synchronization is achieved by exchanging its local time with all channels and adjusting the local clock according to the voted clock. A distributed agreement algorithm is used to establish a global clock and a fault-tolerant voting algorithm is developed to protect the global clock from failure by any type of faults including the Byzantine faults [3].

The RMS employs data voting as the primary mechanism for fault detection, isolation and recovery. If a channel generates data which is different from a voted majority, the voted data will be used as the output to mask the fault. The faulty channel will be identified and penalized by a global penalty system. Data voting includes both application data and system status data. The RMS is designed to support heterogeneous computing systems in which fault-free channels are not guaranteed to produce the exact same data images due to diversified hardware and software. Therefore, the RMS uses a user specified tolerance range to define erroneous behavior should data deviance occur in the voting process.

The RMS supports graceful degradation by excluding a failed channel from a group of synchronized, fault-free channels known as the operating set. A penalty system is designed to penalize erroneous behavior committed by any faulty channel. When a faulty channel exceeds its penalty threshold, other fault-free channels can reconfigure themselves into a new operating set that excludes the faulty channel. The excluded channel is not allowed to participate in data voting and its data is used only for monitoring purpose. The RMS also has the capability, through dynamic reconfiguration, to admit a healthy channel into the operating set. This self-healing feature allows the RMS to preserve system resources for an extended mission.

2.2 System Architecture

A sample three-channel RMS-based system architecture is depicted in figure 2.1. In this architecture, the RMS interconnects three Vehicle Mission Computers (VMC) together to form a redundant, fault-tolerant system. Each VMC has a VME chassis with several single-board computers in it. The RMS is installed in the first board and the communication between RMS and other application boards is through the VME backplane bus. Each VMC takes inputs from its external 1553 buses. The three main applications, Vehicle Subsystem Manager, Flight Manager and Mission Manager, compute their functions and then store critical data in VME global memory for voting.

Figure 2.1 A three-channel RMS-based fault-tolerant system

Each RMS board takes the data via the VME bus and broadcasts the local data to other channels through the CCDL link. After receiving three copies of dada, the RMS will vote and write the voted data back to the VME global memory for use by the applications.

System Fault Tolerance

To describe the RMS fault-tolerance capability, we first need to define the concept of Fault Containment Region (FCR). An FCR usually has a territory bounded by natural hardware/software components. The key property of the FCR is its capability to prevent fault and error propagation into another region. In the RMS design, each channel is defined as a fault containment region for fault detection, isolation, and recovery. Multiple faults occurring in the same region are viewed as a single fault because other regions can detect and correct the fault through voting. The number of simultaneous faults a system can tolerate depend upon the number of fault-free channels available in the system. For non-Byzantine faults, N=2f+1 where N is the number of fault-free channels and f is the number of faults. If a system is required to be Byzantine safe, N=3fB+1 where fB is the number of Byzantine faults.

The RMS is designed to tolerate faults with different time durations such as transient faults, intermittent faults and permanent faults. A transient fault has very short duration and occurs and disappears randomly. An intermittent fault occurs and disappears periodically with a certain frequency. A permanent fault remains in existence indefinitely if no corrective action is taken. In fault tolerant systems design, rigorous pruning of faulty components can shorten fault latency, thus, enhance systems integrity. Nevertheless, immediate exclusion of a transiently faulty component may decrease system resources too quickly and jeopardize mission success. The RMS allows a user to program its penalty system in order to balance these two conflicting demands according to application requirements. Different penalties can be assigned against different data and system errors. High penalty weights for certain faults will result in rapid exclusion of faulty channels should such faults occur. Vise versa, low penalty weights against other faults will allow a faulty channel to stay in the system for a while so that it can correct its fault through voting.

3. RMS Implementation

The RMS has two subsystems, Fault Tolerant Executive (FTE) and Cross-Channel Data Link (CCDL). Furthermore, the FTE consists of five modules: Synchronizer, Voter, Fault Tolerator, Task Communicator, and Kernel. Their functions and operations are described in the following sections.

3.1 System Synchronization

The Synchronizer (SYN) establishes and maintains channel synchronization for the system. It is required that, at any time, each individual RMS must be or operate in one of the five states: POWER_OFF, START_UP, COLD_START, WARM_START, and STEADY_STATE. Figure 3.1 is the state transition diagram of an individual RMS and its five states are described as follows:

POWER_OFF is a state when the RMS is non-operational and the power source of the associated computer is off for any reason. When the RMS is powered-up, the RMS unconditionally transitions to START_UP.
START_UP is the state after the computer has just been powered-up and when all system parameters are being initialized, RMS timing mechanism are being set up and the inter-channel communication links (i.e., CCDL) are being established. When the start-up process is complete, the RMS unconditionally transitions to COLD_START.
COLD_START is the state in which the RMS cannot identify an existing Operating Set (OPS) and is trying to establish an OPS. An OPS is a group of nodes participating in normal system operation and voting.
WARM_START is the state in which the RMS identifies the OPS containing at least 2 RMSs but the local RMS itself is not in the OPS.
STEADY_STATE is the state when the node of the RMS is in synchronization with the OPS. A STEADY_STATE node can be in or out of the OPS. Each node in the OPS is performing its normal operation and voting. A node out of the OPS is excluded from voting but its data is monitored by OPS nodes to determine its qualification for readmission.

Figure 3.1 RMS state transition diagram

In the Cold Start, an Interactive Convergence Algorithm [1] is used to synchronize channel clocks into a converged clock group known as the operating set. All members are required to have a consistent view about their memberships in the operating set and they all switch to the Steady-State mode at the same time.

In the Steady-State mode, each channel broadcasts its local time to all channels through a System State (SS) message. Every channel dynamically adjusts its local clock to the global clock in order to maintain system synchronization. Since RMS is a frame-synchronized system, it has a predetermined time window called the Soft-Error Window (SEW) that defines the maximum allowable synchronization skew. Each fault-free RMS should receive other SS messages in the time interval bounded by the SEW. Since RMS is used in a distributed environment, using a single SEW window has inherent ambiguity in determining synchronization errors among participating channels [2]. To resolve the ambiguity, another time window known as the Hard-Error Window (HEW) is used in the RMS. If channel A receives channel B clock outside of A's HEW, channel A reports a synchronization error against channel B. However, if channel B sees that its own clock (after receiving its own SS message) is in the HEW, channel B reports that channel A has a wrong error report regarding the synchronization. The ambiguity of mutually accusing channels needs to be resolved by other channel's views about channel B's clock. If channel A is correct, other channels should observe that channel B's clock has arrived at least outside of their SEW. Sustained by other channels error reports, the system can then identify channel B as the faulty channel. Otherwise, channel A is the faulty channel because of its deviance from majority view in the error report.

Warm Start is half way between Cold-Start and Steady-State. A channel may be excluded from the operating set because of faults and errors. The excluded channel can go through reset and try to resynchronize with the operating set in the Warm-Start mode. Once the channel detects that it has synchronized with the global clock of the operating set, it can switch to the Steady-State mode.

3.2 System Voting

In RMS, voting is the primary technique used for fault detection, isolation and recovery. The RMS Voter (VTR) votes on system states, error reports and application data. The voting of system states establishes a consistent view about system operation such as the membership in the operating set and synchronization mode. The voting on error reports formulates a consensus about which channel has erroneous behavior and what should be the penalty for the errors. The voting on application data provides correct data output for the application to use. The data voting sequence is shown in figure 3.2.

The RMS data voting is a cyclic operation driven by a minor frame boundary. A minor frame is the period of the most frequently invoked task in the system. As demonstrated in figure 3.2, a four-channel system generates application data in a minor frame and stores the data in a shared memory known as the Application Data Table for RMS to vote. At the minor frame boundary, the Task Communicator (TSC) module of the RMS uses the Data Sequence Table (DST) as pointers to read the data from Application Data Table. The DST is a data voting schedule which determines which data needs to be voted in each minor frame and it also has other associated information necessary for the voting. After reading the data, the TSC packages the data into a certain format and sends the data to the CCDL. The CCDL broadcast its local data to other channels while receiving data from other channels as well. When the data transfer is completed, the Kernel (KRL) takes the data from the CCDL and stores the data in the Data Copies Table where four copies of data are ready for voting. The Voter (VTR) performs voting and deviance checks. A median value selection algorithm [1] is used for integer and real number voting and a majority voting algorithm is used for binary and discrete data voting. The data type and its associated deviance tolerance are also provided by the DST which is used by the VTR to choose a proper voting algorithm. The voted data is stored in the Voted Data Table. At a proper time, the TSC module reads the data from the table and writes them back to the Application Data Table as the voted outputs. Again, the addresses of the output data are provided by the DST. For each voted data, a data conflict flag may be set in the Data Conflict Table by the VTR if the system has only two operating channels left and the VTR detects the existence of data disagreement. The Data Conflict Table is located in a shared memory space so that the application software can access the table to determine if the voted data is valid or not.

Figure 3.2 Data voting sequence diagram

3.3 Fault Tolerator

By the definition of a Fault Containment Region, a FCR can manifest its errors only through message exchanges to other FCR's [3]. Through voting and other error detection mechanisms, the Fault Tolerator (FLT) summarizes errors into the 15 types shown in table 3.1. A 16-bit error vector is employed to log and report detected errors. The error vector is packaged in error report message and broadcast to other channels for consensus and recovery action at every minor frame.

Table 3.1 Error vector table

The FLT assesses a penalty against a channel which is the source of errors. At every minor frame, all detected errors are assigned with penalties and the penalty sum is stored in an Incremental Penalty Count (IPC). The VTR module votes on the IPC and the voted result is stored in a Base Penalty Count (BPC). The IPC captures errors for a particular minor frame and the BPC captures cumulative errors for entire mission time. The BPC is also voted every minor frame and the FLT uses the voted BPC to determine whether a system reconfiguration is required. The Interactive Consistency Algorithm [1] is used for penalty assignment and voting in order to ensure a consistent action among all fault-free channels for system reconfiguration.

The system reconfiguration includes both faulty channel exclusion and healthy channel readmission. If the Base Penalty Count of a faulty channel exceeds a predetermined threshold, the RMS starts the system reconfiguration. The reconfiguration process begins at the boundary of a major frame which is the period that all tasks repeat their operation. During the reconfiguration, the system regroups the operating set to exclude the faulty channel. Once a channel loses its membership in the operating set, its data and system status will no longer be used in voting process. The excluded channel needs to go through a reset process. If the reset process is successful, the channel can try to resynchronize itself with the operating set and it can switch to the Steady-State mode if the synchronization is successful. An excluded channel can operate in the Steady-State mode, but it is still outside of the operating set. The channel now receives all system messages and application data from the nodes in the operating set. However, it will not execute any application tasks nor will its views about other nodes be accepted by the members in the operating set. All members in the operating set also receive messages from the excluded channel and monitor its behavior. The Base Penalty Count of the excluded channel may be increased or decreased depending upon the behavior of the channel. If the excluded channel maintains fault-free operation, its BPC will be gradually decreased to below a predetermined threshold. At the next major frame boundary, the system goes through another reconfiguration to readmit the channel.

3.4 RMS and Application Interface

The current RMS implementation uses the VME bus and shared memory as the RMS and application interface. However, this is only one possible implementation and other communication protocol can also be employed to implement the interface. The main function of the TSC module is to take data from designated communication media and package data into a certain format for RMS to use. When a voting cycle completes, the TSC takes the voted data and sends the data back to application.

RMS Kernel

The Kernel provides all of the supervisory operations for the RMS. The Kernel manages the startup of RMS, calling the appropriate functions to initialize the target processor as well as the loading of all initial data. During the startup process the Kernel configures the CCDL module by loading the system configuration data and the proper operational parameters. The Kernel manages the transitions between the RMS operating modes (i.e. Cold-Start, Warm-Start and Steady-State) by monitoring the status of other RMS modules and taking the appropriate actions at the correct times. The Kernel uses a deterministic scheduling algorithm such that all 'actions' are controlled by a self-contained time base. At a given 'tick' in the time-base cycle, the predetermined actions for that tick are always executed. RMS activities, such as fault detection, isolation and recovery, are scheduled by the Kernel at the appropriate times in the RMS minor frame. If a RMS channel becomes faulty, the Kernel has the responsibility for restarting the channel at the proper time. All data transfers between the RMS subsystems and between RMS and the application computer(s) are managed and scheduled by the Kernel. The Kernel directs the other modules to prepare various RMS messages and loads those messages into the CCDL for transmission at the Kernel's request. As messages are received by the CCDL the Kernel extracts those messages and dispatches them to the correct module(s) for processing. The Kernel runs in a loop, continuously executing each of the scheduled actions and monitoring the RMS status.

Cross Channel Data Link (CCDL)

The CCDL module provides data communication among channels. The data is packaged into messages and the message structure is shown in figure 3.3. Each CCDL has a transmitter and up to eight receivers. The CCDL top-level architecture, Transmitter and Receiver schematics are depicted in figure 3.4 - 3.6. Both horizontal and vertical parity bits are attached to data messages in order to enhance communication reliability. Message format is verified by the CCDL and only valid messages are sent to the Kernel for further processing. Since the CCDL is the only module which establishes physical connection among channels, it must enforce electrical isolation in order to guarantee Fault Containment Regions for the system. In our CCDL design, electrical to optical conversion is used to convert electrical signals to optical signals. In this way, every channel has its own power supply and all of them are electrically isolated from each other.

Figure 3.3 CCDL message structure

Figure 3.4 CCDL top level architecture

Figure 3.5 CCDL transmitter

Figure 3.6 CCDL receiver

4. Claims

We recommend the following claims to establish AlliedSignal's proprietary rights in this invention:

We claim a method that can separate redundancy management from applications.

We claim a set of techniques that make such separation feasible.

We claim the Redundancy Management System based architectures for fault-tolerant applications.

We claim a dynamic system reconfiguration technique that protects system integrity and provides self-healing capability.

We claim a data structure that is used to control the RMS operation.

We claim a method that establishes and maintains system consensus as well as consistency for data processing in a distributed system with existence of any type of fault.

We claim a method and a set of techniques to perform fault-tolerant cross channel (or node) data communication.

We claim a method and a set of algorithms and techniques to perform fault tolerant, frame-based system synchronization.

We claim a method and a set of algorithms and techniques to perform fault-tolerant data voting and deviance checking.

We claim a method and a set of techniques to perform fault/error isolation, detection and recovery

We claim a method and a set of techniques to coordinate system communication, synchronization, voting, fault/error detection, isolation and recovery, and application interface functions

REFERENCE
1. Real Time Executive Module Design Document, AlliedSignal Aerospace
2. P. Thambidurai, A.M. Finn, R.M. Kieckhafer, and C.J. Walter, "Clock synchronization in MAFT," Proc. IEEE 19th International Symposium on Fault-Tolerant Computing, 1989, pp 142-149.
3. J. Zhou, "Design Capture for System Dependability," Proc. Complex Systems Engineering Synthesis and Assessment Workshop, NSWC, Silver Spring, MD , July 1992, pp 107-119.

From Vaporware to Software
The Relational Avionics Planning Tool for Operational Requirements (RAPTOR)

Raymond Szymanski
U.S. Air Force Research Lab (AFRL)
Customer Requirements Branch (SNZC)
2241 Avionics Circle, Suite 1
Wright-Patterson A.F.B., Ohio 45433-7303

ABSTRACT

At the 1997 Joint Avionics Weapons Systems conference (JAWS '97) the author presented a paper entitled "Do You Dare Reuse Software Without Extensive Testing." That paper defined the Air Force's new technology planning activity, the Air Force Modernization Planning Process (AFMPP), and the need for information systems that could support that activity. It described one such information system, RAPTOR, that was created by the author through software reuse and discussed the resulting pluses and minuses of a reuse approach. Finally, it laid out plans for future enhancements to both the existing information system and the tools used for application development.

This paper happily reports and assesses the progress that has been made in the RAPTOR development since JAWS '97. It will cover the rational for RAPTOR's development, present details on RAPTOR's main features, discuss traps and pitfalls encountered during the development and how to avoid them, and finally, present a look-ahead to the next generation RAPTOR. One of the major events in the RAPTOR development was to scrap its FoxPro implementation and replace it with a version implemented in Microsoft Access and Visual Basic 5. The rational for replacing this development environment is discussed along with near-term and far-term benefits realized as a result of the change.

1 Who Needs RAPTOR?

1.0 Customer Requirements Branch

The Customer Requirements Branch resides in the Sensors Directorate (SN) of the Air Force Research Laboratory at Wright-Patterson Air Force Base. Its mission is to facilitate the transition of Sensor Directorate technology to the operational user. To accomplish this mission the SNZC engineers must first gain a full understanding of Air Force operational needs, a full understanding of SN-developed technology, and a full understanding of the inter-relationships between the needs and the technologies. This understanding manifests itself in an on-paper linkage between the operational needs and the technology, and establishes a focussed list of potential customers. After the customers have been defined, SNZC needs to bring together the technology developers and the potential customers. This action is necessary to initiate a dialogue between the participants to determine the interest level from both sides and establish formal cooperative agreements. Lastly, SNZC engineers must facilitate the exchange of comprehensive, accurate, and timely information between the technology suppliers and the users, on a regular basis, to ensure a successful technology transition.

All SNZC activity takes place within the framework of the Air Force Modernization Planning Process (AFMPP). In the simplified process discussed above, many of the required communication and information activities and participants in those activities that facilitate the AFMPP are not discussed. However, all in SNZC agree that its main product is information and the manipulation of information into relative formats and useful data for a vast array of AFMPP participants to use and leverage.

1.1 Information Hunters and Gatherers

The information that SNZC collects and distributes is wide and varied in both content and sensitivity. On the collection side SNZC has numerous suppliers of information. The Air Force Mission Area Teams supply mission area operational needs in their annual or biannual Mission Area Plans. Sensor Directorate engineers supply Investment Strategy Data Sheets that document current and future Advanced Technology Demonstrations (ATDs). SN engineers and AFRL associated contractors supply Concept Solutions that address operational deficiencies. SN program managers supply data on their technical programs including white papers and briefing charts.

On the information distribution side SNZC has more participants and customers than on the collection side. Rather than acting as a mere information 'pass-through' SNZC either value adds to the information it collects or filters the information to enhance its usability by the information customer. Recent information-intensive activities that SNZC has supported include the Office of the Secretary of Defense's (OSD) Technology Area Review and Assessment (TARA) and Defense Technology Area Plan (DTAP), the Space Program Objective Memorandum (POM) build, and the Aeronautical Systems Center's (ASC/XR) Concept Call. Unfortunately, each activity required manual integration of information from disparate sources; activities that could have been greatly simplified with the proper support tools.

1.2 Beyond the Big 3

If Andy Rooney, the venerable philosopher of "Sixty Minutes", were to ponder the question "Why were EXCEL, PowerPoint, and Word invented?" he might respond thusly. "So we can keep lots of good information bottled up in big packages and spend lots of time looking for a little fragment in those packages when we need it. At the conclusion of the recent TARA, DTAP, Space POM, and Concept Call activities the participants offices strongly resembled the disheveled look of the Mr. Rooney's own workspace. After tearing through hundreds of files to rack, stack and relate information its no surprise that the offices in question were in need of a quick bulldozing. What would have been useful is a database that contained much of the necessary information with facilities for presenting the information in the required format. That type of help is now available.

In March 1998, the first installation of the RAPTOR database was accomplished in the Customer Requirements Branch. RAPTOR was immediately put to the test as the office manager input Advanced Technology Demonstration (ATD) data. When the tasked was finished the comment made by the user and heard 'round the office was "It was so easy I could have done it with my eyes closed". Grown men wept.

2. RAPTOR

2.0 Overview

RAPTOR Version 1.0 is a database developed in Microsoft Access 97 and Visual Basic 5.0 that addresses the SNZC primary task of " information and the manipulation of information into relative formats and useful data for a vast array of people to use and leverage". RAPTOR runs under Windows 95 and was designed as a single user application that can handle sensitive information. With it the users can collect, connect, view, administer, control, and distribute fine-grained information as it relates to a number of selected 'primary data areas' and 'secondary data areas'.

The primary data areas have strong ties to the AFMPP and include Mission Areas, Needs, Concepts, Subsystems, Tech Needs, Programs, ATDs, Points of Contact (POCs), and NonATDs. The secondary data areas complement the primary data areas but differ in scope since they are document-oriented and provide a big picture view of resident data rather than a granular one. Secondary data areas include Documents and Reports.

Although RAPTOR can perform all the functions listed above, its real strength is its capability to relate or link the various data that it contains. Using this capability, one can create new information that can be stored and viewed within RAPTOR, printed to hard copy, or easily exported to a file for further manipulation and analysis. Now, instead of data in separate files related only by the file name, fine-grained data is explicitly related and exploitable into new information.

2.1 Data Collection

RAPTOR supports two principal means of data collection, one for the RAPTOR administrator and one for the typical user. Figure 2.1 shows the form that is the user's interface for viewing, collecting, managing, and linking data. This form is divided into tabs with one tab for each primary and secondary data area. To move from tab to tab the user merely clicks on the tab of interest. Figure 2.1 illustrates how RAPTOR's main form appears when the Concepts tab is selected and the form is ready to accept a new concept.

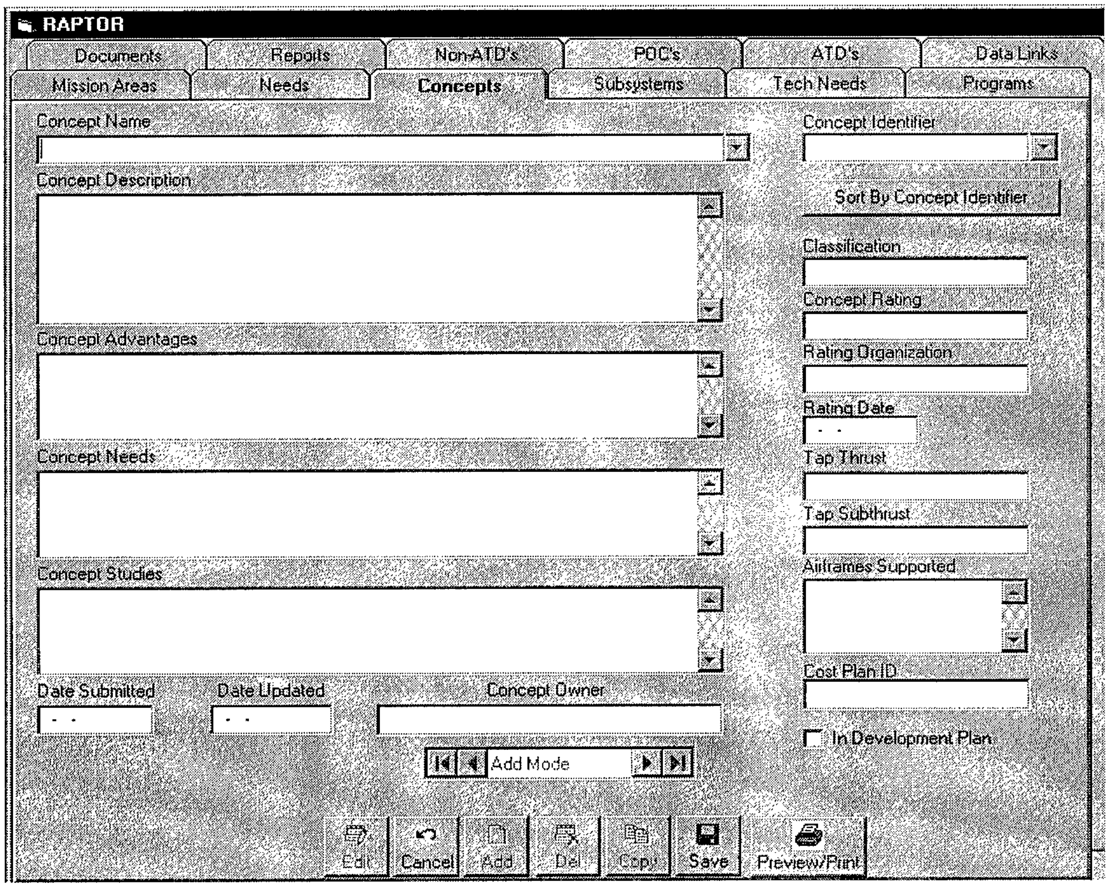


Figure 2.1 RAPTOR Main Form - Concepts Tab

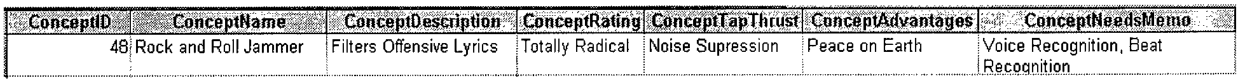
To enter a new record the user must first put the desired tab in the Add Mode. This is accomplished by selecting the Edit button and then the Add button. The form is now ready to accept data into any one of the 18 data boxes seen on the Concepts tab. Data boxes are selected by directly clicking inside of the desired box or by tabbing to it. Once the cursor is in the desired data box the user can type text directly into it or paste text from the clipboard. Data that has been entered is saved to the underlying database by clicking on Save. Selecting Cancel will discard any data that has been entered on the form and will return the tab to Browse Mode. The same procedure is followed for inputting data to any of the primary data area tabs.

Records do not have to be entered one at a time if the desired information is already in electronic columnar format. Since the database administrator has complete access to the underlying database tables, he can append large chunks of data, comprising multiple records, to the tables directly. These large chunks of data do not have to be in Access format as the Access tools will perform format conversion prior to appending new records.

2.2 Data Connections

2.2.1 Within and Between Records

Each data box in Figure 2.1 represents a field in the underlying Concepts data table. Data entered in a data box is stored in the appropriate field. Figure 2.2 shows seven of the nineteen fields from the Concepts table. The first row of Figure 2.2 reveals the actual field names used within the table. The second row contains data that was entered into the database via the Concepts tab. This data is all part of a single record and by definition is related or connected to the other data within that single record.



ConceptID	ConceptName	ConceptDescription	ConceptRating	ConceptTapThrust	ConceptAdvantages	ConceptNeedsMemo
48	Rock and Roll Jammer	Filters Offensive Lyrics	Totally Radical	Noise Supression	Peace on Earth	Voice Recognition, Beat Recognition

Figure 2.2 Concepts Table - Partial Single Record

The capability to relate data is the essential element in the process of making useful information from scattered data. To this end, RAPTOR facilitates data relationships both within and between records. Any record from any of the principal data areas can be explicitly related to any other record in RAPTOR. The capability to relate any record in RAPTOR to any other record provides the user with virtually unlimited ways to exploit the resident data.

Figure 2.3 illustrates the complete set of connections that can be made between RAPTOR data records. If there is a line in Figure 2.3 between a data area and another data area then RAPTOR permits linking of records between those areas. This scheme supports one record-to-one record and one record-to-multiple records relationships.

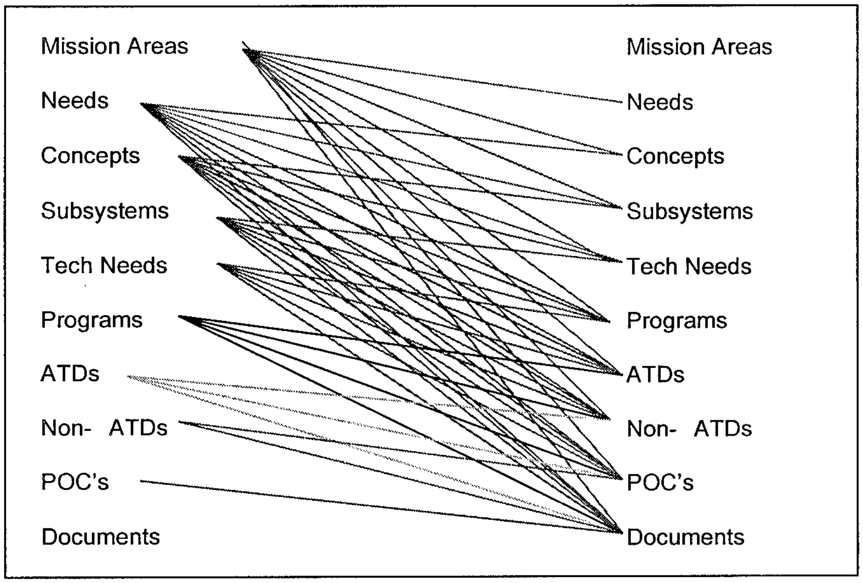
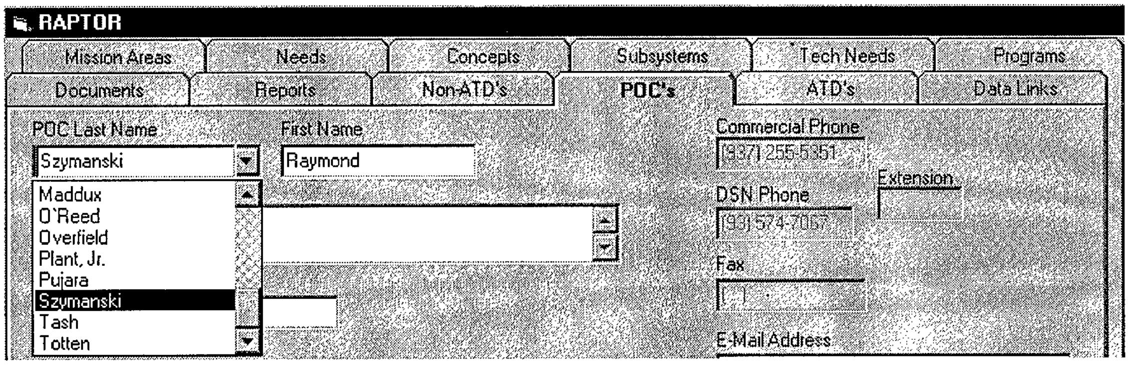


Figure 2.3 Data Area Connection Possibilities

2.2.2 Making Connections

As the following example illustrates, making connections between RAPTOR records is as easy as pointing and clicking. In this fictitious example, the user is the Chief of the Mission Requirements Branch. He wants to assign his employees responsibility for the individual Mission Areas. Because a particular employee was late for the last staff meeting he will be assigned responsibility for five Mission Areas.

To begin the assignment process the user selects the POCs data tab to access the employee's record. An employee is selected by using the POCs Last Name dropdown list box that displays the employees' names. Figure 2.4 shows the upper section of the POCs tab along with the POC Last Name dropdown list. After a selection is made from the list, the selected employee's record is displayed. The Chief may now link this employee record to as many Mission Areas as are available within the database.



POC Last Name: Szymanski

First Name: Raymond

Commercial Phone: (937) 255-5351

Extension:

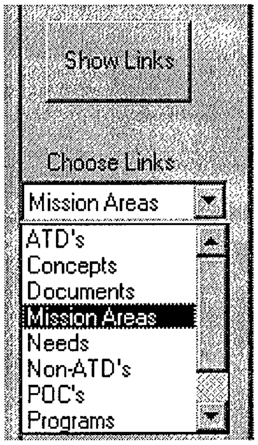
DSN Phone: (93) 574-7067

Fax: ( ) -

E-Mail Address

Figure 2.4 POC Tab - Last Name Dropdown List

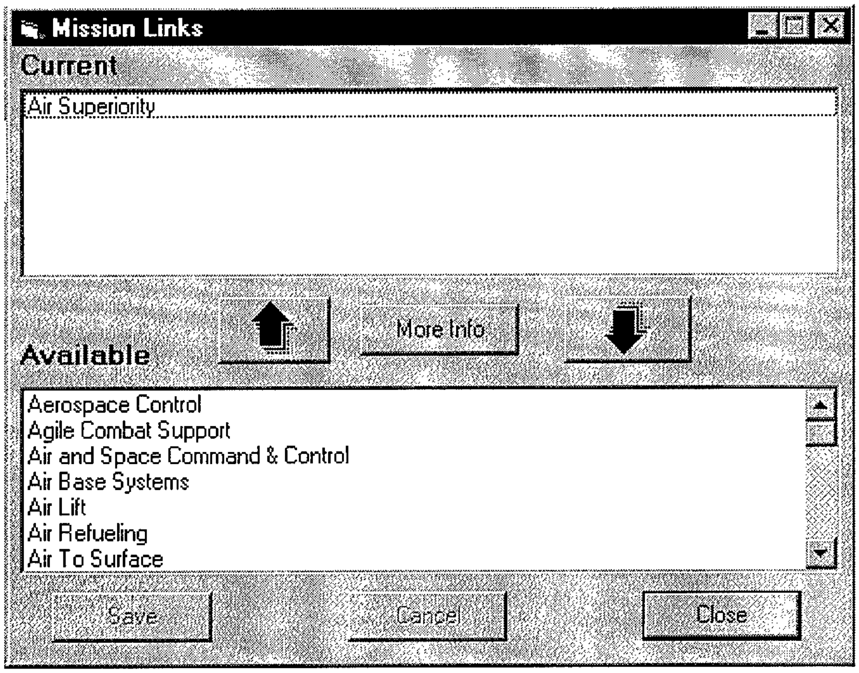
Since the user wants to make links between the selected employee and Mission Areas he chooses 'Mission Areas' from the 'Choose Links' dropdown list and then clicks on the Show Links button (Figure 2.5). These actions cause the 'Mission Links'



Choose Links: Mission Areas

Figure 2.5 Choose Links/Show Links

dialogue box to be displayed (Figure 2.6). The dialogue box is used to make the actual links between the selected employee and the desired Mission Areas.



Air Superiority

Aerospace Control
Agile Combat Support
Air and Space Command & Control
Air Base Systems
Air Lift
Air Refueling
Air To Surface

Figure 2.6 Mission Links Dialogue Box

Figure 2.6 shows that the Air Superiority Mission Area is already assigned to the employee by virtue of its appearance in the Current list. The Available list contains the name of every Mission Area in RAPTOR that is not currently assigned to the selected employee. To assign additional Mission Areas, the user must move the desired items from the Available list to the Current list and then save the links by choosing the Save button. Items are moved from one list to the other by either double clicking on the item or by selecting one or more items in a list and then clicking on the appropriate arrow found between the Current and Available lists.

2.3 Data Viewing and Navigation

2.3.1 Single Records

RAPTOR facilitates data record viewing and navigation through multiple data navigation mechanisms and views. The first view the user becomes familiar with is the Tab View that was seen in Figure 2.1 RAPTOR Main Form - Concepts Tab. Recall that this view exposes all the fields for a single record on a single tab. There are two ways to navigate data records in this view, sequentially forwards or backwards and directly.

Figure 2.7 shows the Record Navigation Control that appears on every data area tab. Clicking the right inner arrow will cause the tab to display the next record in that sequence. Clicking the left inner arrow will cause the tab to display the previous record

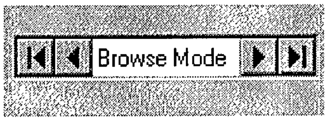


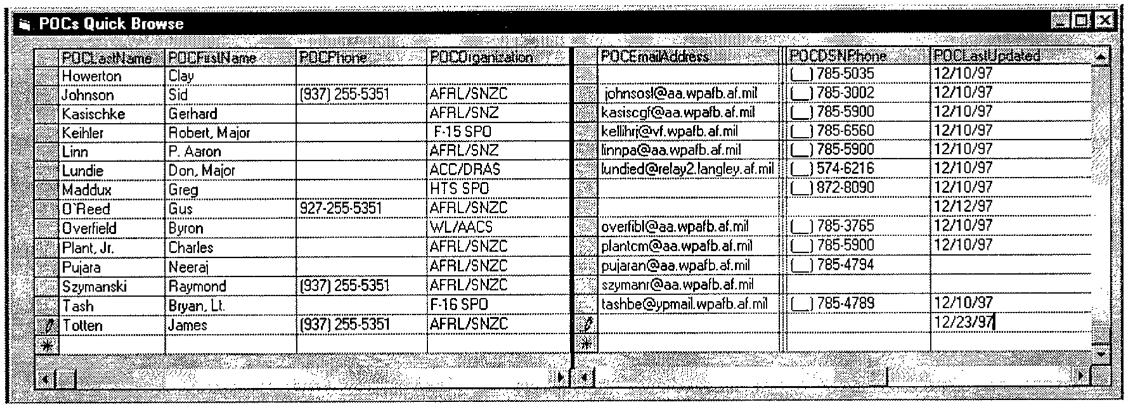
Figure 2.7 Record Navigation Control

in that sequence. Before RAPTOR displays its records in Tab View, it first sorts them in alphabetical order by the Name field. Therefore, the inner arrows select either the previous or next record alphabetically by Name. The left and right outside arrows are used to display the first record or the last record respectively in the alphabetized record set. The Record Navigation Control is the most efficient method of sequential record navigation within RAPTOR.

In section 2.2.2, Making Connections, The POC Last Name dropdown list box was used to directly select the employee's record for assignment to Mission Areas. To facilitate direct selection of records for each data area there is a similar Names dropdown list box. If the user were on the Programs tab, for example, choosing the Program Names list box would produce a list of program names contained in RAPTOR. Choosing from that list causes the display to show the complete record of the program that was selected from the list box.

2.3.2 Multiple Records

Each data area tab is associated with a recordset. This recordset contains every data field for every record for the selected tab. To view a recordset for a particular tab the user must first select the tab and then click the Give Me Data button. Figure 2.8 shows the data grid that is displayed when the Give Me Data button is selected while on the POCs tab. Similar data grids are available for each data area tab in RAPTOR.


POCs Quick Browse

POCLastName	POCFirstName	POCPhone	POCOrganization	POCEmailAddress	POCDSNPhone	POCLastUpdated
Howerton	Clay				(__) 785-5035	12/10/97
Johnson	Sid	(937) 255-5351	AFRL/SNZC	johnsosl@aa.wpafb.af.mil	(__) 785-3002	12/10/97
Kasischke	Gerhard		AFRL/SNZ	kasiscgf@aa.wpafb.af.mil	(__) 785-5900	12/10/97
Keihler	Robert, Major		F-15 SPO	kellihrj@vf.wpafb.af.mil	(__) 785-6560	12/10/97
Linn	P. Aaron		AFRL/SNZ	linnpa@aa.wpafb.af.mil	(__) 785-5900	12/10/97
Lundie	Don, Major		ACC/DRAS	lundied@relay2.langley.af.mil	(__) 574-6216	12/10/97
Maddux	Greg		HTS SPO		(__) 872-8090	12/10/97
O`Reed	Gus	927-255-5351	AFRL/SNZC			12/12/97
Overfield	Byron		WL/AACS	overfibl@aa.wpafb.af.mil	(__) 785-3765	12/10/97
Plant, Jr.	Charles		AFRL/SNZC	plantcm@aa.wpafb.af.mil	(__) 785-5900	12/10/97
Pujara	Neeraj		AFRL/SNZC	pujaran@aa.wpafb.af.mil	(__) 785-4794	
Szymanski	Raymond	(937) 255-5351	AFRL/SNZC	szymanr@aa.wpafb.af.mil		
Tash	Bryan, Lt.		F-16 SPO	tashbe@ypmail.wpafb.af.mil	(__) 785-4789	12/10/97
Totten	James	(937) 255-5351	AFRL/SNZC			12/23/97

Figure 2.8 POCs Data Grid

The data grid provides an effective means of viewing and navigating an entire data area tab recordset. Vertical scroll bars permit sequential navigation through the recordset while horizontal scroll bars permit viewing each field in the wider-than-the-screen record. To further aid in data viewing, the data grid provides the user with user-adjustable field height and width and data table-splitting capabilities.

Figure 2.8 is an example of a data grid that employs the data table-splitting capabilities. Since there are fifteen data fields for each record in the POCs table, their columns cannot be easily displayed at one time. This becomes a problem if the areas of interest happen to be the first few columns of a wide table and the last few columns in the same table. The split table data grid solves this problem by permitting the independent horizontal scrolling of the left and right tables of the data grid. The left table and right table records always stay in synchronization because there is but a single vertical scroll bar to control both table views.

In Figure 2.8 the user is viewing four fields from the leftmost side of the POC table and three fields from the its rightmost side. Upon close examination one can find two additional vertical lines between the columns labeled POCEmailAddress and POCDSNPhone. These lines represent two columns from the POC data table that were horizontally collapsed by the user to bring the Email and DSNPhone columns next to each other for easier viewing.

2.4 Data Reporting and Distribution

2.4.1 Unlinked Data

In the previous sections on data viewing it is evident that what you see is what you entered, basically, data in record format. In data reporting the view has to be more sophisticated and must exploit the data to create information. After all, that's supposed to be the primary reason for using a database.

RAPTOR uses Crystal Reports as its principal report generation and distribution package. The Crystal Reports report generation system that comes bundled with the Visual Basic 5.0 development environment is easy to use and is highly integrated with Visual Basic 5.0. This makes it an ideal reporting system for VB application developers.

RAPTOR provides two principal entry points to the built-in predefined reports. The first is found by clicking on the Print/Preview button. This action produces a dialogue box that offers opportunities to preview and print a number of reports related to the current data area tab. Figure 2.9 shows the tab-oriented print preview dialogue box.

The second report generation entry point is found by selecting the Reports tab. The reports that can be generated from this form are not dependent on a singular tab but can generate reports for any RAPTOR data area. This form also includes the capability to generate reports that reflect the data linking activity discussed in an earlier section. Figure 2.10 shows the Reports tab report-selection form.

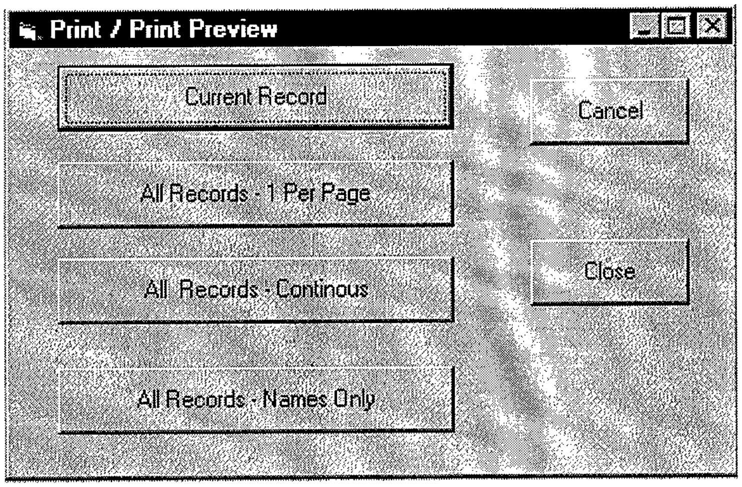


Figure 2.9 Print/Preview Dialogue Box

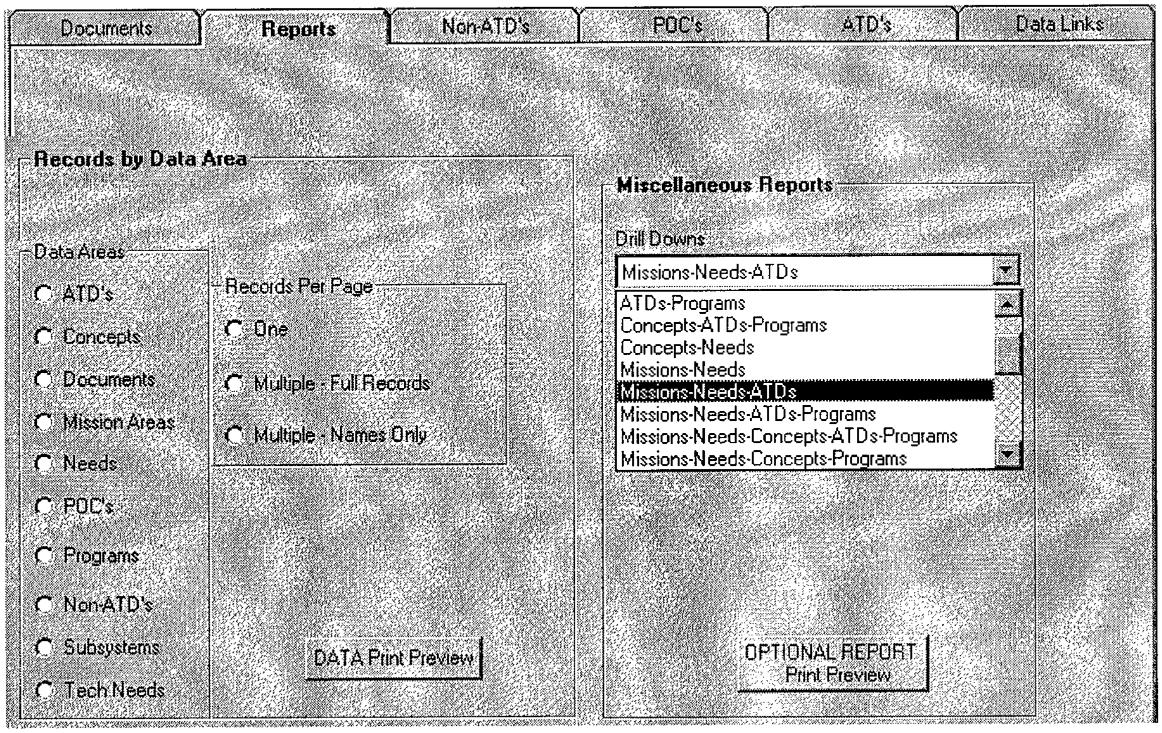


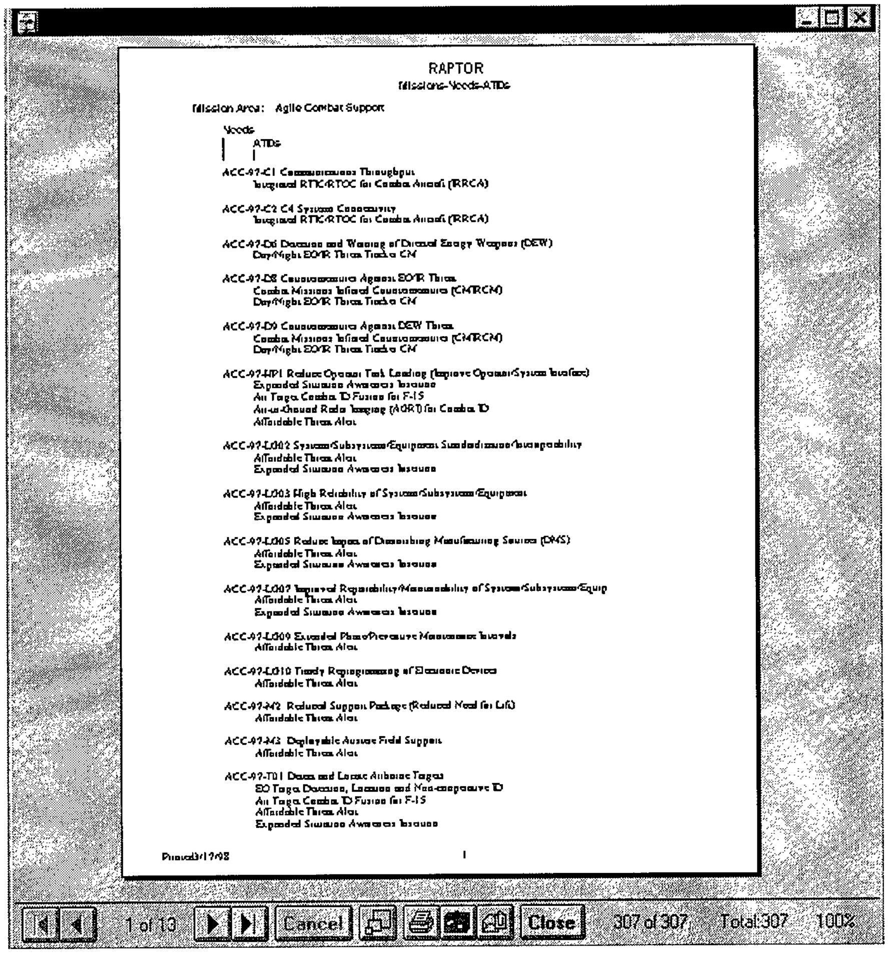
Figure 2.10 Reports Tab

The reports generated by RAPTOR have been designed to meet the requirements of SNZC's diverse customer base. However, it is not always possible to anticipate future customer-information requirements. Fortunately, Crystal Reports provides the capability to export any RAPTOR report to a file. This is an important capability for the user who needs to either customize a report to meet specific customer requirements or to forward the report information in electronic format. If necessary, the user can rely on RAPTOR to provide data linking, initial data sorting and filtering, and then publish the report in whatever format is required.

2.4.2 Linked Data

On the right side of the Figure 2.10 Reports Tab, is a dropdown list box that contains the names of several predefined RAPTOR Drill Down reports. These reports are designed to show the relationships between data areas that were established by the user during the linking process. Since RAPTOR permits any data area record to be linked with any other data area record the number of possible unique reports that can be generated numbers in the hundreds. To keep the reports manageable and keep the report writer from quitting his job, the reports listed in the drill down list box were limited to fulfill current SNZC customer and employee requirements. This list can be readily expanded in the future to include new reports as their requirements are defined.

Figure 2.11 presents the Mission Area-Needs-ATDs drill down report that the user selected from the dropdown list and then generated by selecting the Print button.



RAPTOR
Missions-Needs-ATDs

Mission Area: Agile Combat Support

Needs
ATDs

ACC-97-C1 Communications Throughput
Integrated RTIC/RTOC for Combat Aircraft (IRRCA)

ACC-97-C2 C4 Systems Connectivity
Integrated RTIC/RTOC for Combat Aircraft (IRRCA)

ACC-97-D6 Detection and Warning of Directed Energy Weapons (DEW)
Day/Night EO/IR Threat Tracker CM

ACC-97-D8 Countermeasures Against EO/IR Threat
Combat Missions Infrared Countermeasures (CMIRCM)
Day/Night EO/IR Threat Tracker CM

ACC-97-D9 Countermeasures Against DEW Threat
Combat Missions Infrared Countermeasures (CMIRCM)
Day/Night EO/IR Threat Tracker CM

ACC-97-HP1 Reduce Operator Task Loading (Improve Operator/System Interface)
Expanded Situation Awareness Insertion
Air Target Combat ID Fusion for F-15
Air-to-Ground Radar Imaging (AGRI) for Combat ID
Affordable Threat Alert

ACC-97-L002 System/Subsystem/Equipment Standardization/Interoperability
Affordable Threat Alert
Expanded Situation Awareness Insertion

ACC-97-L003 High Reliability of System/Subsystem/Equipment
Affordable Threat Alert
Expanded Situation Awareness Insertion

ACC-97-L005 Reduce Impact of Diminishing Manufacturing Sources (DMS)
Affordable Threat Alert
Expanded Situation Awareness Insertion

ACC-97-L007 Improved Repairability/Maintainability of System/Subsystem/Equip
Affordable Threat Alert
Expanded Situation Awareness Insertion

ACC-97-L009 Extended Phase/Preventive Maintenance Intervals
Affordable Threat Alert

ACC-97-L010 Timely Reprogramming of Electronic Devices
Affordable Threat Alert

ACC-97-M2 Reduced Support Package (Reduced Need for Lift)
Affordable Threat Alert

ACC-97-M3 Deployable Austere Field Support
Affordable Threat Alert

ACC-97-T01 Detect and Locate Airborne Targets
EO Target Detection, Location and Non-cooperative ID
Air Target Combat ID Fusion for F-15
Affordable Threat Alert
Expanded Situation Awareness Insertion

Printed3/12/98 1

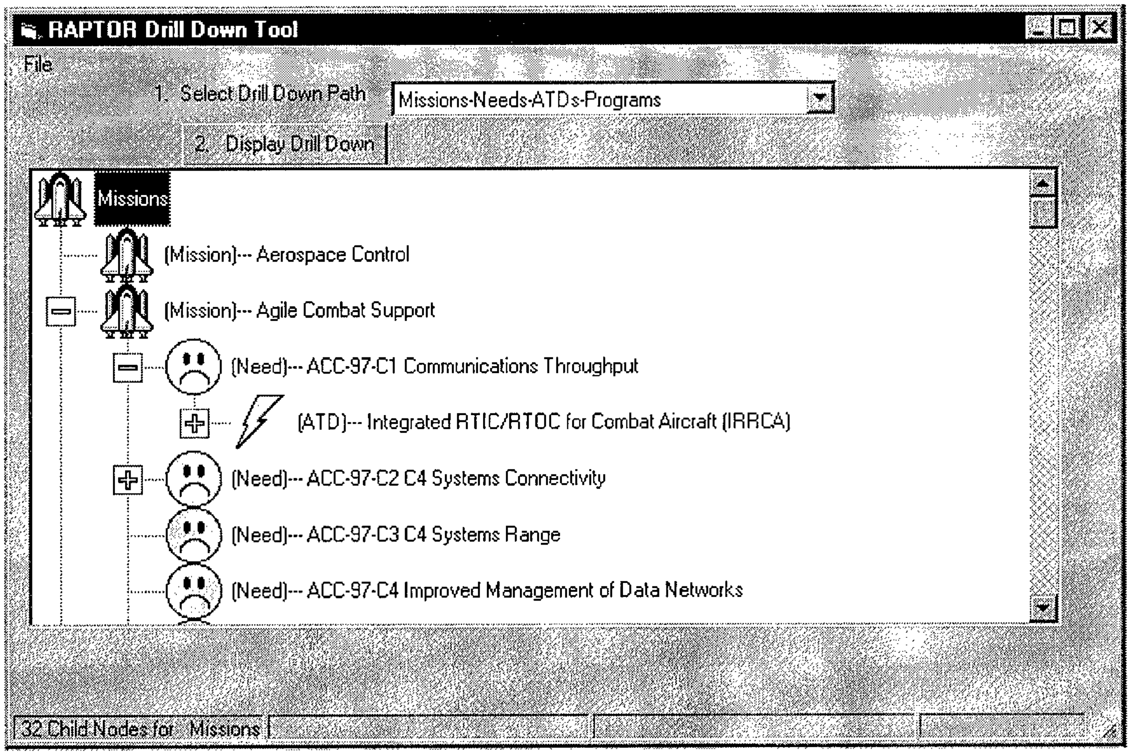
Figure 2.11 Mission Area-Needs-ATDs Drill Down Report Example

This report examines the operational Needs for each Mission Area and reports on those Needs that are linked to a supporting ATD. By design the report automatically page-breaks as the Mission Area changes and indents the Needs and ATDs data to improve readability. All of the built-in drill down reports follow the format displayed in Figure 2.11. However, as revealed in Figure 2.10, the reports vary in the number and type of data areas they report on.

The lower portion of Figure 2.11 shows the navigation and activity buttons that Crystal Reports provides to output the reports and move from report page to report page. Crystal Reports also assists the user by providing some modestly interesting statistical information each time it presents a report preview. In the lower left corner of the Crystal Reports screen it shows the user how many pages are required to print the report and which of those pages is currently displayed. In the lower right corner it reports on the number of unique records that comprise the current report.

2.5 Data Analysis

Although many of RAPTOR's reports can be used to analyze the contents of the database, there is one mechanism that was specifically designed for that purpose, the RAPTOR Drill Down Tool (DDT). While it is very similar to the drill down reports just discussed, even using the same database queries to retrieve its data, it is also very different. The key difference between the drill down reports and the DDT is that the DDT provides the information in a tree view that the user can navigate.



1. Select Drill Down Path: Missions-Needs-ATDs-Programs

Missions
(Mission)--- Aerospace Control
(Mission)--- Agile Combat Support
(Need)--- ACC-97-C1 Communications Throughput
(ATD)--- Integrated RTIC/RTOC for Combat Aircraft (IRRCA)
(Need)--- ACC-97-C2 C4 Systems Connectivity
(Need)--- ACC-97-C3 C4 Systems Range
(Need)--- ACC-97-C4 Improved Management of Data Networks

32 Child Nodes for Missions

Figure 2.12 RAPTOR's Drill Down Tool (DDT)

Figure 2.12 shows the DDT dialogue box that is accessed by selecting the Drill Downs command button.

As an analysis tool the DDT is particularly useful for locating and isolating specific information chains resident in the database. This capability is extremely important to those responsible for populating the database with both information and data links. Remembering once again that database strength is predicated upon storing and relating data, the DDT can help the user quickly ascertain the depth and breadth of information and links contained in RAPTOR. This point will be illustrated in the following simple example.

Having been recently assigned responsibility for four new Mission Areas, the now repentant employee activates the DDT with the Mission Areas-Needs-ATDs-Programs selection. The DDT displays thirty-two space shuttle icons, one for each Mission Area. A quick vertical scroll reveals that one of the employee's newly assigned Mission Areas is not listed, two others do not have any Needs assigned to them, and the final one has assigned Needs but nothing else. This rapid analysis is possible because the icon-oriented DDT provides look-ahead to the next level of detail with a navigable tree structure that is already familiar to any Windows user. The look-ahead capability is provided through the use of '-' and '+' symbols as seen in Figure 2.12. The '-' indicates that a level has already been fully expanded while a '+' indicates that there is at least one additional level of detail available.

In the above example the user could have obtained the same information through the built-in reports or data viewing capabilities but not with the same ease as provided by the DDT. The user quickly realizes he needs to get the missing data into RAPTOR and vows never again to be late for another staff meeting.

3.0 Lessons Learned (Things your mother never told you)

3.1 FoxPro vs. Access/Visual Basic

In 'Do You Dare Reuse Software Without Extensive Testing' the author presented a number of reasons for changing RAPTOR's development environment from FoxPro to a combination of Access and Visual Basic. Foremost among the reasons given was Visual Basic's graphical development environment that "fosters accurate, rapid prototyping and full system development support". Although this statement proved true for the author, several months of development experience with the Visual Basic environment allows one to see that there is much, much more. In fact, comparing VB to Foxpro is like racing a finely tuned Lamborghini against a 1965 Volkswagen bus loaded with hippies; ride the bus and you'll be left in the dust.

Based on extensive experience using both development environments, it is now the author's opinion that the FoxPro environment is obsolete for new Windows-oriented application development. Therefore, there is no merit in providing a feature-by-feature comparison of FoxPro to Access/Visual Basic in this paper. Anyone possessing minimal experience with both environments would certainly render the same opinion. For the skeptical, I recommend your study begin with some light reading in the form of the Foxpro Language Reference Manual that checks in at a meager 1,227 pages.

3.2 Access

It is entirely possible to develop robust desktop databases using Access as the exclusive development environment. This environment is light years ahead of FoxPro in both functionality and user friendliness with its graphical user interface and plethora of Wizards to automate mundane tasks. In fact, RAPTOR would have been developed exclusively in Access had the VB environment not been available with its added features.

After extensive experimentation with both Access and VB and consultation with respected experts, each development tool was applied to tasks that they are best suited to. Access was used to develop RAPTOR's underlying data tables and to develop the database queries that bring data to the user interface. VB was used to create the user interface and to produce the event-driven code.

Creating a table in Access is as easy as typing into a spreadsheet. Linking completed tables together is as easy as point, click and drag. In spite of this, it is also easy to unnecessarily slow the application development and occasionally confound yourself if you do not completely and accurately define the necessary data and link tables first! Before creating any data retrieval queries or any user interface forms, you must be absolutely sure that you are perfectly satisfied with the names that you have assigned to each data field in every table. Why? Because if you change a field name in a table that has been used in a query or form and do not recreate the query or adjust the form with the new name you will experience critical errors at the most inconvenient of times during the application's execution.

Access is not too particular about what you name your tables, what you name the fields in the tables, or what you name the queries. Sure, names are limited to 64 characters, must begin with a letter, can contain numbers, can't contain punctuation marks, and a few other not-too-restrictive rules. This flexibility can be extremely useful or self-defeating depending on whether or not you follow a consistent naming convention for all the items in you application. Highly recommended is the use of the object naming conventions found in the VB Books Online 'Naming Conventions-Coding' help files. Why use VB naming conventions for Access objects? Good question!

The VB naming conventions differ slightly from those enforced by Access but should be used to avoid conflicts when building Access/VB applications. For example, while Access allows the use of spaces in their names, VB does not. VB code limits names to 40 characters while Access permits 64. Code a VB reference to an Access table whose name contains a space or is over 40 characters long and you guessed it, boom! Use the VB naming conventions and the resulting programs and their objects will be easy to read and understand without cramping the programmer's natural creativity. Don't use these and watch programmer productivity drop about 25% as they continually try to guess what type of objects they're working with and where all the strange error messages are coming from.

3.3 Visual Basic

Visual Basic applications begin life as a control object known as a form. Other VB control objects, such data controls text boxes and command buttons are added to the form to facilitate data transfer between the form and the underlying data tables. All of these objects have properties that define the object's physical and state attributes. For example, the Name property provides a unique identifier for each object that can be used to refer to that object in code. Other properties such as Width and Height define a control's two-dimensional display size. Each object also has events associated with it. When an event occurs for an object, like clicking on the object, it responds in a manner defined in that object's event procedure code. Objects also have methods that affect the entire object when invoked. A good example is the Hide method of a form that removes the form from view and reduces it to an entry on the status bar.

The seasoned VB programmer knows that properties, methods, and events vary from object to object although there is some commonality. However, even the veteran programmer can get caught off guard if they assume that two objects having the same properties or methods will react in a similar manner or will acquire the same state. Take the case of a simple text box and a masked edit box. Both controls can display and receive data linked to data tables. When used for data input the masked edit box can also enforce a character-oriented template or filter, restricting the data that it will accept. If the user desires to disable both controls to prevent data input there is a very noticeable resultant surprise on the user's screen. While the text in the text box is clearly displayed, as it was before the disable action, the text in the masked edit box is grayed-out. Surprise!

In many cases these little surprises are not identified in the literature and are popularly referred to as undocumented features. For those situational caveats that are documented they are generally found at the bottom of the help file data sheet for the control object. It is critical to read the entire data sheet for the control along with the respective property, event and method documentation before committing it to a project. After reading the documentation, experiment with the control under near-real project conditions to determine if your understanding of a control's functionality matches reality. Yes, its fun to just jump in and start slapping controls on a form to get your application to do something. However, the time spent fully studying the controls far offsets the time spent ripping out controls that can't quite do what you need them to do when you need them to do it.

Visual Basic applications are event driven. Click a command button and the code in that button's Click_Event will execute. The command button, like all other VB objects, has a name property which the user should set using the naming conventions mentioned earlier. To avoid losing the code and being confronted with one of the most confounding error messages in VB, do not put any code behind a VB object until you are absolutely sure you will not change the name of the object. What happens if this advice is ignored is illustrated below.

A command button named cmdClose will have "Private Sub cmdClose_Click ()" as the first line of its click event procedure. This line is automatically generated by VB. If you add code to this procedure and then rename the button cmdExit, VB will generate another click event procedure opening line, " Private Sub cmdExit_Click ()". Unfortunately, the cmdClose code does not follow the button after you rename it. When you click on the newly renamed command button, cmdExit, it will do nothing and you will be scratching your head at the resultant error message.

Changing the name of an object after putting code in one of its events is bad. Changing the name of an object that has been referenced by another object's code is worse. Executing code containing a reference to an object that no longer exists will return 'Runtime Error 424, Object Required'. The more objects that have been referenced in the offending code, the more fun it is to fix this delightful faux pas.

The number of creative ways to cause trouble by changing the names of objects after they have been referenced by another object is bounded only by one's imagination. Here is one last classic. A Data control links text boxes and other controls to data tables and query-based recordsets. Two Data control properties must be set for this linkage to occur, DatabaseName and RecordSource. To display data from the Data control's recordset two textbox properties must be set, DataSource and DataField. DataSource is set equal to the Name property of the Data control while DataField is set to a field name returned by the Data control. If you change the name of the Data control AFTER setting the DataSource to the Data control's original name, the corresponding text box will look at you like a deer in the headlights. As a matter or fact, if you set any of the above four properties to legal object names and then change the names of those objects, you will get a variety of unexpected inactivity or system error messages.

Quite honestly, there were a large number of other development environment surprises encountered that could be reported here but for the sake of space are not. Of these the most important is the interaction or lack thereof between certain control objects that you want to work together. In one instance two controls, A and B, were chosen to work as a team. Control A would return a recordset while control B would expose the data to the user. Unfortunately, control A could not return the type of recordset that was required for control B to work properly. Once again, to avoid this type of heartache, read all there is to read on a control before betting your application's success on it.

4.0 Future Enhancements

RAPTOR is currently a standalone single-user system. To enable everyone in SNZC to work with their own copy and synchronize data with every other user, an Access facility called replication is used. In replication there is a single master database controlled by the administrator and multiple replicas, one for each user. Changes to the replicas are synchronized with the master, one replica at a time. When all the replicas have been synchronized, new replicas with the cumulative data are distributed. This process will suffice until late fall 98 when SNZC acquires its own server and a client-server version of RAPTOR is deployed.

One of the data area tabs not previously discussed is the Data Links tab. This tab is the future entry point to a number of other local databases that contain information related to data in RAPTOR. The Plans and Programs (P&P) database which tracks programmatic information is certainly of interest to SNZC. Rather than recreating the P&P information in RAPTOR, it will link to the P&P tables over the local area net and extract the data as required. There are other databases that RAPTOR will be cooperating with and similar arrangements for this cooperation are currently under investigation.

The built-in report generation capability is currently limited to the two dozen or so predefined reports. This capability will be significantly enhanced through the use of a third party tool, English Wizard. Reports can be generated based on English-like questions that the user formulates and the tool turns into actual database queries. Although reports can be generated using Access's report generator or VB's Crystal Reports, English Wizard will be easier to use for those who do not care to learn the gory details behind the database's table structure.

As this paper is being prepared, RAPTOR is being installed throughout SNZC. These users are expected to render extensive constructive criticism and suggestions for improvements that will be seriously considered by the developer. The short-term plan is to immediately fix any bugs that are found and redistribute the repaired application. The long-term plan is to incorporate major enhancements into the client-server development. The developer hopes that this approach will keep the dogs from the door and at the same time give the users what the developer wants, the all-important lust for the upgrade.

References

1. Szymanski, Raymond, 'Do You Dare Reuse Software Without Extensive Testing?', 'Joint Avionics, Weapons, and Systems S3 Conference', June 1997.

2. Microsoft Corporation, 'Visual Basic Programmer's Guide', Appendix B - Visual Basic Coding Conventions, 1997.

Biography

Mr. Szymanski is currently the Air Superiority Technical Planning Integrated Product Team representative for the Air Force Research Laboratory, Sensors Directorate, Customer Requirements Branch. He is also responsible for managing the development of the Branch's information technology infrastructure He holds a BSEE from the University of Detroit, did his graduate work in Computer Engineering at Wright State University, and is a certified Windows Application Developer. He is Level III certified in Systems

Planning, Research, Development and Engineering. His technical programs have won two of the prestigious Avionics Technology Transition of the Year Awards. During his tenure as Evaluation & Validation (E&V) Program Manager he chaired the Ada Joint Program Office's E&V Advisory Team, served on the Ada Federal Advisory Board, and was a member of the KAPSE Interface Development Team (Mil-Std-1838A). Mr. Szymanski has lectured at the Naval Postgraduate School in Monterey and the Air Force Computer Resource Acquisition School at Brooks AFB, Texas. He has authored papers for many technical symposiums both domestic and abroad, including the Portable Common Tool Environment (PCTE) conference last held in Paris. His recently completed assignment in the Plans and Programs office as the directorate's technical representative culminated in his receipt of the 1996 Laboratory Excellence Award for Process Improvement. When he is not slinging code or writing papers he enjoys playing golf along with his sons and coaching youth baseball and soccer.

Using RMA Throughout an Avionics Product Development Lifecycle

By Peter Kortmann
Tri-Pacific Software
Alameda, California USA
(510) 814-1770
Email: peter@tripac.com
http://www.tripac.com

Fast and Efficient Software

- Dichotomy in the world of real-time systems development, especially in the spacecraft engineering community.
- Systems need to be "Fast", after all they are real-time.
- To get them "Fast" we get the fastest processor we can afford to use tempered by power considerations and environmental factors.

Fast and Efficient Software
(Continued)

- Then we get the software to run "efficiently.
- "Efficient" means to get the system software to run quickly and to run "deterministically".
- Efficient interpreted as not engineering the software according to modern software engineering principles.

Handcrafted Software

- The concepts of encapsulation, abstraction, and modularity are often abandoned in the name of machine cycle efficiency.
- Determinism also drives another nail into the coffin of good software engineering.
- We need to know where every byte of the software physically resides, exactly how an answer is determined to stimuli in a cycle by cycle manner.

Handcrafted Software
(Continued)

- We do not use commercial software components because there is danger in not having the source code and everyone knows that commercial software is full of bugs!
- So we handcraft it all according to a methodology that is not much different than those used in the days of the 4-bit microcontroller.

Achieving Performance Costly

- The end result is a piece of software that works and works very well.
- The downside is that this performance has been achieved at great expense.
- Software now "spaghetti" code with global data areas, machine dependencies, and a hand engineered execution timeline of major and minor cycles.

Achieving Performance Costly
(Continued)

- Everything is hand-crafted.
- It is the equivalent of a tailor made suit, except that in addition to cutting the material for the jacket and trousers, we found it necessary to weave the material from raw wool.

More Functionality, Same Price

- Even with newer and faster spacecraft processors still same old process.
- Except now customers expect much more functionality for the same price.
- To provide those functions, but be cost-effective we must use off the shelf solutions.

More Functionality, Same Price
(Continued)

- But how do we integrate these solutions into an integrated real-time software system?
- Integrating commercial products into a major/minor type of scheduler without knowing how the commercial products work and without the source code is extremely difficult and fraught with risk.
- So how do we do it??

Rate Monotonic Analysis and Scheduling is the Key!

- Rate Monotonics enables us to engineer a real-time system expeditiously and realistically.
- It is used throughout the system engineering and software engineering cycle.
- We want to take the guesswork out of our design evolution.

Rate Monotonic Analysis and Scheduling is the Key! (Continued)

- We want to evaluate timing concerns realistically rather than just assuming that a certain type of interface or operating construct is slow.
- Oftentimes we find that certain constructs are quite expeditious when taken in the system context.

Keys to Successful Implementations

- We also want to get away from the handcrafted notion of real-time systems software engineering.
- We want to implement modern characteristics of software engineering such as encapsulation, modularity, abstraction, etc.

Keys to Successful Implementations
(Continued)

- Get rid of the global data definitions that lead to the creation of spaghetti code.
- Software needs to be simple, easy to understand, and stable under worst case conditions.
- Here are the keys to a successful implementation.

How to Use RMA/RMS and Applicable Support Tools on Real-Time Spacecraft Projects

- **<u>Step 1) System Engineering.</u>**
- First - system requirements decomposed into as complete a set of "transfer functions" as possible.
- The transfer function set should explicitly state what happens in the system, how it reacts, and what the transfer produces.

Step 1) System Engineering.
(Continued)

- The transfer functions adhere to the event table entries in the RMA handbook from the Software Engineering Institute (SEI).
- What we do not have at this point are the stimuli rates, these are next.
- However before that, the definition of the transfer function set can get to be pretty heated!

Step 1) System Engineering.
(Continued)

- Most staff know what the system is supposed to do, but decomposing it into discrete transfer function entities is often a bone of contention.
- However, this process is invaluable to establish a common baseline for discussion of system timing requirements.

Step 1) System Engineering.
(Continued)

- Once we get the set of transfer functions defined, we need to establish the criteria for initiating the work each is to perform.
- This is where system-timing issues start to be explored in a systematic manner.
- It is also where hardware design specifications start to meet the realities of the software architecture.

Step 1) System Engineering.
(Continued)

- Timing is the key to a successful system.
- Hardware and software must meld to yield a cost effective system solution.
- The kind of issues discussed usually start at the level of stochastic process modeling for the system environment.

Step 1) System Engineering.
(Continued)

- These issues include such things as:
  - how many messages per second we can expect in the comm uplink,
  - how many targets must we track per scan period,
  - what vibration rate can we expect coming through eclipse in the case of structural damping.

Step 1) System Engineering.
(Continued)

- What is the worst case probability of a component error and what is the maximum recovery time?
- These issues can get contentious, however it is better to resolve them early in the design cycle where change is effected more cheaply.
- What also comes out of this stage is a set of operational modes for the system and usually the first evolution of transition criteria from mode to mode.

Step 1) System Engineering.
(Continued)

- At this point we know what we expect the system to do, how often we expect the system to do it, an idea of any synchronization requirements that may exist between the transfer functions, and a set of modes and modal transitions.
- We have not yet begun implementing a design…..we just described the real world model of what we want to do!

Step 2) Decompose into a System Design

- Now that we know what to do and how often to do it, we can now start to design a system.
- This is best modeled as an iterative process.
- We design, model, test, measure, adjust, and re-design.
- We continue to do this until we get something that schedules and is stable!

Step 2) Decompose into a System Design (Continued)

- We also make an assumption that we will use a RTOS that supports some if not all Rate Monotonic Scheduling constructs.
- We assume that this system is doing more than can be humanly engineered with major/minor cycles such as lan/wan control, target processing, attitude determination and control, all on the same processor or set of processors.

Step 2) Decompose into a System Design (Continued)

- In some commercial software such as middleware, concurrent task daemons may be implemented.
- This pushes us to utilize a multi-tasking concurrent thread model of processing.
- We don’t try to modify large commercial middleware packages into a traditional major/minor cycle paradigm.

Step 2) Decompose into a System Design (Continued)

- This is tough if we do not have the source code.
- We instead work with their design to support the easiest integration effort.
- The more packages we wish to integrate the more pliable we must become.

Step 2) Decompose into a System Design (Continued)

- Luckily most of the commercial packages that execute under VxWorks are schedulable in an integrated system using RMA.
- Now that we have our transfer function set for each system mode, we treat each mode separately and further refine the timing requirements for the system relative to the transfer function sets.

Step 2) Decompose into a System Design (Continued)

- For each transfer function we need:
  -) Stimuli Rate (and Jitter rates if applicable): From System Engineering Stage.
  -) Processing time.
  -) Deadline for processing to be complete.
- Stimuli rate was addressed at a high level in the system-engineering phase.
- Need to be more detailed level here to characterize the stimuli arrival pattern, etc.

Step 2) Decompose into a System Design (Continued)

- Processing time is the time we expect the algorithm triggered by the stimulus to execute (e.g. Work to be done).
- In early stages of the design this is often a ball park estimate until a benchmark on the target hardware can be performed.
- The deadline is identical to the definition of a cycle overrun in a traditional model.

Step 2) Decompose into a System Design (Continued)

- Missing a hard deadline is considered an error.
- RMA allows us to expand this definition for our design to accommodate pre-period deadlines and post period deadlines.
- Explanations of these types of deadlines are beyond this simple tutorial.

Step 2) Decompose into a System Design (Continued)

- At this point we also take a first crack at the layout of the software system.
- What type of data pools are required, when do the transfer functions (now considered concurrent threads of control) synchronize, how long does a thread require a system resource such as a memory area or disk drive?

Step 2) Decompose into a System Design (Continued)

- How many tasks are used to construct a thread?
- These issues bring up the question of how we best utilize the features of the real-time operating system (RTOS) to build our system.
- Fortunately an RTOS such as VxWorks provides features that directly support the precepts of RMS.

Step 2) Decompose into a System Design (Continued)

- These are features such as priority inheritance semaphores or fixed priority - preemptive task execution algorithms.
- Even early stages of design can provide an idea of how such a system can be built realistically.
- We have our threads, task breakdown, stimuli rates…..this allows us to define the priorities for all of our tasks according to RMA principles.

Step 2) Decompose into a System Design (Continued)

- The RMA model phase also allows us to assign the proper priority level to hardware interrupts at this stage.
- Interrupts are merely stimuli into the system and like all stimuli have a defined arrival pattern.
- These are also scheduled according to RMA principles.

Step 2) Decompose into a System Design (Continued)

- We just think of the ISR's as tasks having a higher priority than the application level defined tasks and a priority assigned relative to the stimuli rate of the other interrupts.
- ---- Now we iterate, iterate, iterate until we get a system that is stable and schedulable.

Step 2) Decompose into a System Design (Continued)

- What comes out of the iteration is an identification of performance bottlenecks, issues of algorithm efficiency, issues of CPU bandwidth.
- This information allows us to perform system level tradeoffs for CPU selection on the basis of our system design model rather than on a uncertain model based upon MIPS or other synthetic benchmark number.

Step 2) Decompose into a System Design (Continued)

- Basically at this stage we have a definite model we can adjust and evaluate with engineering discipline without having to yet commit to the expense of prototyping a system.
- Assuming we get a system model that can schedule, we now can implement the model.

Step 3) Implementation

- The most important aspect of a real-time system is the timing!
- Without solid timing on the target we cannot guarantee schedule and without schedule guarantees we do not truly have a real-time system.
- It may be really fast, but not real-time!

Step 3) Implementation
(Continued)

- Building a traditional major/minor cycle based system we often implement the algorithms into the cyclic model and evaluate performance in a series of builds.
- We are really combining the system timing aspects with the transfer function processing at this stage.
- This is a serial process : implement the scheduler, put in some algorithm, measure performance, test.

Step 3) Implementation
(Continued)

- But not very efficient.
- Using the RMA approach we can perform system timing studies on a target in parallel with algorithm measurement and put the two together later.
- For implementation of timing, we need only to take our RMA model designed to utilize RTOS constructs that support RMA and implement the timing portion of our design.

Step 3) Implementation (Continued)

- This includes presenting stimuli from the external world, passing of data between tasks, designing access to common resources using the correct semaphores, etc.
- We do not need the actual algorithm at this stage, only a dummy load within each task that mimics the use of processor bandwidth.
- We now have a skeleton of our system implemented on the target without the guts of the system algorithms.

Step 3) Implementation
(Continued)

- Assuming we implement this system under VxWorks, we can use the WindRiver WindView tool set to evaluate the system level task timing and performance directly.
- We have now entered another iterative design stage.
- We have implemented, executed, measured, and now we adjust for performance.

Step 3) Implementation
(Continued)

- As a bonus we also get a realistic look at the dynamics of the operating system overhead at a very early stage of our development cycle.
- This O/S overhead is not constant but is variable based upon how we have utilized the O/S constructs to implement our system.
- As we adjust, this overhead may vary, sometimes by a critical amount!

Step 3) Implementation
(Continued)

- We can now handle this variability because we can directly see it using WindView.
- In parallel to the run time portion of the project, the application engineers are coding up the algorithms required to fill up the guts of the tasks now using dummy work loads to burn CPU bandwidth.

Step 3) Implementation
(Continued)

- The applications can now be written with respect to the templates presented by the task and synchronization mechanisms implemented in the timing model.
- Since each thread also has a deadline associated the applications staff now also knows how much time they have to complete their execution cycle.

Step 3) Implementation
(Continued)

- This portion is done in parallel using the same operating system and target and measuring performance with WindView.
- If done correctly we can then drop the application into the task model without much trouble.
- We have now partitioned application development from detailed timing and sped up the implementation of the project.

Step 3) Implementation
(Continued)

- We can now utilize a small cadre of highly skilled real-time system engineers to implement the timing model while we use engineers skilled in applications development to build the algorithms.
- Real-time systems engineers are usually in a shorter supply than applications developers and are therefore more expensive.

Step 3) Implementation
(Continued)

- Hopefully we can decrease our overall staffing costs for the project.
- We can still not forget our RMA model developed prior to our implementation phase.
- Fortunately the WindView tool can present a detailed set of real-time measurements directly to the PERTS (Prototyping Environment for Real-Time Systems) RMA toolset.

Step 3) Implementation
(Continued)

- Using the WindView data, PERTS will construct a model of the system that can be compared to the original analytic model.
- The analytic model is then calibrated to the performance model.
- Once this is done we can take advantage of the calibrated model to evaluate “what-if” scenarios late in the project stage due to various changes.

Step 3) Implementation
(Continued)

- We can evaluate the impact of proposed requirement changes for technical, schedule, and cost impact, evaluate proposed modifications to hardware that manifest in stimuli or synchronization changes, etc.
- The RMA model has now become the living template of our system performance.

Step 4) Once We Delivered the Project

- Once the system is complete and delivered, the RMA model is still a very valuable entity.
- The tradition is to archive our source code, executable, COTS tools, etc. into a configuration management system for possible later use.

Step 4) Once We Delivered the Project (Continued)

- We have documented our system and archived the specifications and design documents.
- What is often missing is a living model detailing the timing nature of the system.
- This is where we archive the RMA model.
- Invaluable for fielding modifications of our originally delivered system.

Step 4) Once We Delivered the Project (Continued)

- CPU upgrades or customer requested modifications can be evaluated before a costing effort is attempted.
- Archiving the model with the delivered system retains all of the benefits of the model through the design and implementation phase into the life cycle phase.

MODELNET:
A MODELING AND SIMULATION ENVIRONMENT FOR LEGACY PROGRAMS

Dr. John H. Schaibly
Science Applications International Corporation
10260 Campus Point Drive
San Diego, CA 92121
and
John E. Camp
Air Force Research Laboratory / IFSD
2241 Avionics Circle
WPAFB, Ohio 45433

Abstract

MODELNET is a modeling and simulation environment for running legacy codes on UNIX platforms. It
provides a user shell to organize and run a set of related models and exchange data between them. MODELNET assists the model *user* by providing GUI input, visual model linking tools, context sensitive help, integrated documentation, and shared I/O utilities. It assists the model *developer* by organizing model directories, compilation tools, and shared libraries. It supports collaborative engineering environments where model and data sharing are important. While MODELNET components are available at essentially no cost, Science Applications International Corporation (SAIC) offers model integration services and integration training.

Background

Over many years engineering software models have been developed by industry, government and university with varying degrees of quality and utility. We are all familiar with programs that are superbly documented, have a handsome interface, but whose technical capabilities are limited or give wrong answers, and other models which are poorly documented, riddled with untrapped errors, clumsy inputs, but which give highly accurate answers, and, with hundreds or thousands of users, are considered standards.

Each model has a history from concept through marketing of that concept, prototype development, development, delivery and maintenance and user support. Generally, the models which "rise to the top" and are still in widespread use after many years are technically accurate and solve useful, persistent problems. But there are many models abandoned along the way not because of deficiency of quality, but for political, organizational or personal reasons. If a government program is canceled, if a key developer retires, or if a developing company fails, good technical software and technical information disappears or is filed away out of view. Ironically, after a few years have passed, there is often a new requirement in a new program office who funds a new contract to a new contractor to rediscover and redevelop the old technology. Developers who have spent a lifetime in this environment are often painfully aware of how few of their products were widely used and valued. Others take advantage of the quick obsolescence to resell the same technology repeatedly to different customers causing inflated costs and wasted effort.

Ignoring the political, organizational and personal aspects of software obsolescence, there are technical reasons why legacy software ceases to be used. As computer technology progresses, users are reluctant to use programs that seem outmoded. Younger users familiar and trained on software developed in the Age of Microsoft will be impatient and frustrated at older methods of input based on the 80 column punched card or even the menu driven inputs of the 1970's and 80's. They are not comfortable with poorly defined or complicated input instructions. MODELNET addresses these problems.

Where there used to be teams of analysts working on a problem, now fewer, less experienced users are expected to get answers often without the expert advice and oversight present in the past. MODELNET also addresses this problem.

Then there is the multiplicity of models. If more than one legacy model is available for solving a particular problem, the user must choose between them based on the usual speed/accuracy tradeoff. How does the accuracy of model results affect the results of subsequent model calculations using those results as inputs? MODELNET addresses these problems.

Expanding problem domains may require several models to be linked in sequence. Thus data passing between models becomes a real issue. Much effort is spent interfacing data between models. This becomes a difficult and error-prone process when the data source and destination work in differing units, resolutions, or datatype. Often inputs require calculating a combination of outputs from one or more models requiring sequential control over the models. MODELNET addresses these problems.

In modern software engineering, there is an emphasis on model reuse, distributed processing and model decomposition. In some cases, a set of existing sub calculations need to be integrated into a single aggregate model. In other cases, an existing complex calculation requires breaking up into a set of components which may further be required to be reassembled in a new configuration. This aggregation / disaggregation may occur at several levels: procedural level, subroutine level, or in modern software, object level. MODELNET addresses these issues.

Finally, MODELNET supports the collaborative engineering environment. As models are developed by different developers at different locations, even using different platforms, they can share their work easily. The MODELNET development process has sufficient documented standards that guarantees portability to other MODELNET systems running on other UNIX platforms.

MODELNET is not the first attempt at a modeling and simulation environment / system to address these various issues Some of these have come and gone and others are on hold awaiting funding. MODELNET is different in that the framework is made up of components which are either public domain or ubiquitous off the shelf existing products which have life independent of MODELNET; the system can grow and evolve as the components grow.

MODELNET components

MODELNET is a development and user framework; a block diagram is shown in Figure 1. It consists of Khoros 2.2, Browser, CLIPS6 (C Language Integrated Production System) Expert System tool from NASA, and a library of Freeware utilities. By using only freely available components, MODELNET is basically free. By using components which are common off-the-shelf and supported by many thousands of users, MODELNET works today, is maintained by the component developer, and will leverage advancements of these components to advancements in MODELNET. Furthermore, users will see familiar user interfaces and not have to learn yet another one. What users are not familiar with Netscape Browsers, spreadsheet programs, and point and click GUI's? MODELNET uses the components unmodified so that, as new backward compatible versions appear, all implemented models will be compatible.

Khoros 2.2 component in MODELNET

Khoros 2.2 is a major part of MODELNET. It is developed by Khoral Research Inc. in Albuquerque, NM as an image processing environment. It has a graphical programming user interface called Cantata which allows icons ("glyphs") to be moved around the screen using the mouse, much like arranging boxes using a graphics draw program. Each glyph represents a program process. There are handles on the glyph which can be connected to handles from other glyphs indicating that data will flow between them. Figure 2 shows a Cantata screen containing five glyphs representing, from left to right, an input scene image file as generated by GTRENDER, a noise addition process, a jitter process which uses a vibration power density function, and an output viewer. This set of five glyphs is referred to as a "glyph (or model) network". When the "RUN" button is pressed, Cantata starts the first process and when it is finished, the second process starts automatically. Data is passed between the glyphs only when calculation of that data is complete.

In addition to a large library of image processing and image data handling processes included with Cantata, control glyphs can be embedded in the network. For example, there can be "if - then - else" branches, where completion of a process can trigger one of two succeeding processes depending on some user specified or other process specified condition. Repetition loops can be added to repeat sequentially running a network systematically varying user specified parameters. The network has intelligence so that in multiple runs it will not repeat running processes if the inputs have not changed.

In Khoros, integrated program modules are called "objects" (not to be confused with "object oriented"). Each object contains an executable, source code, supporting code for data manipulation, GUI data and help files. One of the Khoros development modules (Composer) sets up development directories, make files, and writes code shells for integration software. Figure 3 shows the modules of Khoros which we use in MODELNET. Included in Composer is a GUI builder and an implicit GUI visual structure which is the same for all integrated models. A set of objects is called a Toolbox which is managed by another Khoros development module called Craftsman. This allows groups of models for specific application areas to be collectively turned on or off, or delivered to different customers, or shared by an authorized community of users.

While Khoros was developed for integrating image processing applications, it has the facility for integrating much more complex applications. The Khoros training courses (which are very good) only deal with software development of simple models and do not address issues arising from complex model integration. SAIC has developed a methodology for integrating such complex models for two toolboxes described below, and the methodology appears to be applicable to a large class of application areas.

Browser component in MODELNET

The browser component brings to MODELNET the capability of hypertext documentation and context sensitive help systems. We have used Netscape, although Mosaic or HotJava browsers work similarly. We have linked Khoros objects and Netscape so that they can call each other. Thus, if a user clicks on the help button of a glyph's GUI, the help page is brought up in the browser window. All the hypertext features of the browser are brought to bear on providing the level of help required. Links to the complete documentation system are possible making a truly integrated system. The documentation can be in HTML accessible by the browser directly, and in PDF, also accessible by most browsers. Since these formats are readily editable, the user can tailor documentation and hypertext to his own needs. Figure 4 shows the uses of the Browser in MODELNET.

If the platform is connected to the internet, all those features are available to the MODELNET system. For example, sources for many kinds of data are then available to MODELNET users. E-mail contact to model developers (presumably for a small fee) is available on the same platform. The browser also provides a client-side interface to other servers for distributed computing applications.

CLIPS component in MODELNET

Figure 5 shows the uses of CLIPS 6.0 in MODELNET. Prototypes for CLIPS integration have been produced, but no delivered examples have been implemented to date. The design for CLIPS integration includes checking input variables, output variables (for obvious errors), and most importantly, intermodel consistency of inputs and outputs to assure compatible units and ranges of validity. The rule database is also be editable by the user.

Multiple Levels of Model Aggregation

MODELNET supports multiple levels of model aggregation depending on user need. Typically, an experienced user must set up a network of model components validating all input variables, and setting up the required databases. However, using a Khoros feature called encapsulation, the set of data and models can be combined into one process with a reduced set of inputs (typically scenario definition variables such as range, altitude. The encapsulated model can then be run easily by a relatively unsophisticated user who is protected from worrying about unfamiliar inputs and processes.

MODELNET also facilitates disaggregation. For example, many engineering codes already contain calls to subprocesses. These are very easy to separate in MODELNET, allowing substitution of alternative models for one or more of the subprocesses. Of course, data translation of inputs and outputs may be required. There is also a trend in the M&S community to componentize existing legacy models, that is, functionally decompose the models into components which are often rewritten as object oriented code. We are also looking at adapting MODELNET to these types of modules as well, but we have not yet demonstrated that capability.

Current MODELNET applications

SAIC has integrated two major toolboxes using MODELNET. One, the Advanced Electromagnetic Model for Aerial Targeting (AEM*AT), developed for AFRL, integrated a set of IR signature and sensor codes for aircraft. Models integrated included SPIRITS 4.2, an IR aircraft signature model; EMBED, a target embedding model for combining target and textured background scenes; LASERX, an active signature model using solids models; EOSSIM, a sensor model, and XPATCH version 1, a radar cross section code. A preliminary integrated version of Georgia Tech's scene generation programs GTRENDER and GTSIG is also included. This simulation can be run in an animation, moving the target and sensor according to a detailed flyout trajectory.

The second toolbox developed by SAIC was NEOTAM, the NATO Electro-Optical Target Model specializing in missile plume signature models developed for the Research

Study Group RSG-18. Models from the participating NATO countries were integrated to provide a common model set for predicting missile signatures.

Evolutionary Integration into MODELNET

The integration procedure SAIC has developed for MODELNET is a multi-step process which allows basic integration with very little effort, and increasing utility with increasing integration. This accommodates customers who have limited resources or who want frequent review and control over the final product. The process may be broken into the following levels:

Level 1 - porting the legacy model to the desired platform and verifying its operation. This can be done either by the integrator or by the customer.

Level 2 - calling the model from the Khoros Cantata workspace as a standalone process; use legacy input method / files and data.

Level 3 - developing a Quick GUI for changing frequent inputs such as scenario variables, or inputs depending on other models already integrated into MODELNET, or input variables for which multiple values are routinely run (batch runs). The user would enter all inputs using the legacy input methodology, and only the variables in the Quick GUI would be changed at network runtime. Quick GUIs consist of only one page of point and click inputs. This level includes writing the software for model data integration.

Level 4 - developing a comprehensive GUI for most of the inputs for the model. This can result in many (tens of) pages of inputs which are organized according to the Khoros GUI paradigm. SAIC has extended the Khoros capability of using either Athena or Motif widgets to include Java based widgets as well. User documentation is also rewritten to accommodate the new GUI inputs.

Level 5 - completing documentation and expert system. This final level will best be completed after the familiar user has been working with the MODELNET implementation of the model and can write the requirements. Hypertext online documentation is provided and integrated with other documentation in the toolbox. The expert system rules are defined by the user and formatted by the developer to assist the inexperienced user and to help all users avoid and correct mistakes.

Levels 1 and 2 integration typically take only a few days. Level 3 requires development of software to parse and write the legacy input file formats. The other levels of integration require correspondingly more effort.

Model and toolbox documentation are done both in HTML and Adobe Acrobat (PDF) formats. Both of these formats are commonplace and allow the user to modify the documentation for his own purposes. Documentation forms can include written reports, color graphics, AVI or QuickTime movies, sound, and internet sources as well as e-mail.

MODELNET Portability

MODELNET applications are developed on either (or both) the Sun or Silicon Graphics workstations. When Khoros becomes available on the NT, our toolboxes can be ported easily to that platform as well. Portability of MODELNET depends on portability of Khoros and the Browser. Application toolboxes developed on one platform can typically be ported to one of the other supported platforms in a day or two. Supported platforms include:

Platform	Khoros	Netscape	CLIPS
Sun (> Solaris 2.4, > SunOS 5.4)	x	x	x
SGI (> IRIX 5.2)	x	x	x
LINUX(PC) (Yggdrasil, Redhat)	x	x	x
FreeBSD, BSD/05 (PC)	x	x	x
IBM RS/6000	x	x	x
HP 9000.(> 700)	x	x	x
DEC Alpha AXP (OSF/1 > 3.0)	x	x	x
DG Aviion (DG/UX)	x	x	x
Macintosh		x	x
DOS		x	x
Windows 95 / NT		x	

MODELNET Availability

MODELNET is a software integration service sold by SAIC to military and industrial customers. Customers who have software which they would like to integrate under this system should contract with the MODELNET development team for all or some of the integration effort. SAIC also encourages teaming arrangements with similar companies on other software integration / development procurements. If desired, SAIC will provide tutorial instruction and continuing user support on all phases of the integration process allowing customers to perform their own integration in the future. If integrated models are of interest to a wider community of users and the customer desires to market them commercially, SAIC can work with the customer to provide a distribution package. The customer will retain possession and distribution rights to any legacy models unless special agreements are made.

Interested parties please contact Dr. John Schaibly at SAIC (john.h.schaibly@cpmx.saic.com) or call him at (619)646-4025 for information on the MODELNET system.

Figure1 - MODELNET Block Diagram.

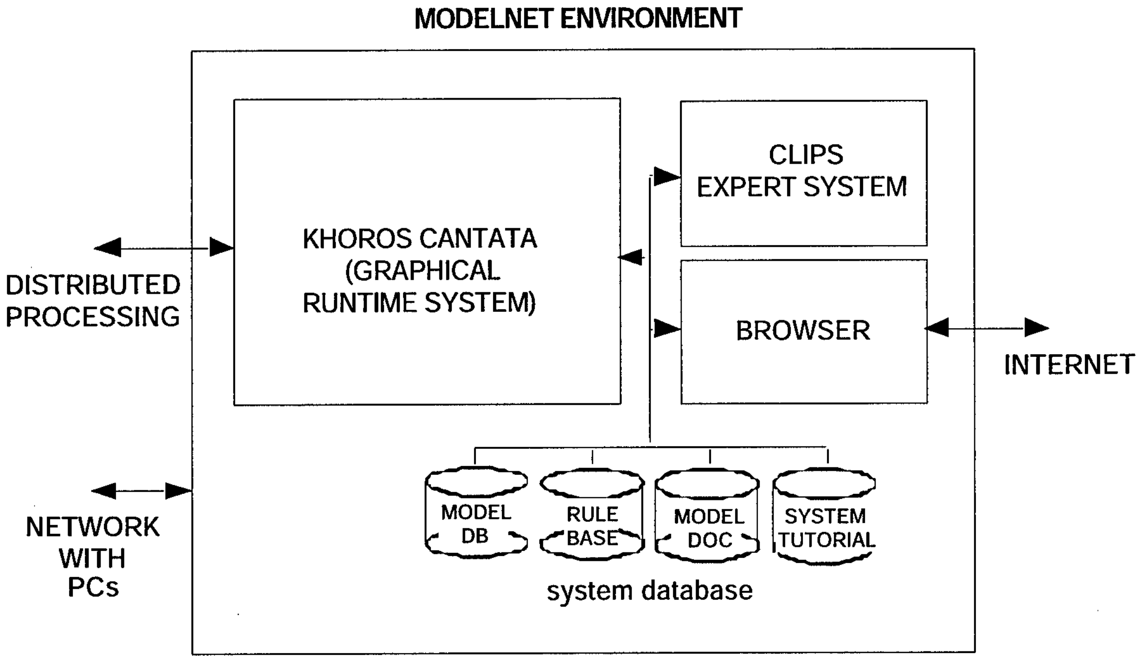


Figure2 - Example of the Khoros Cantata Visual Programming Interface for a Sensor simulation.

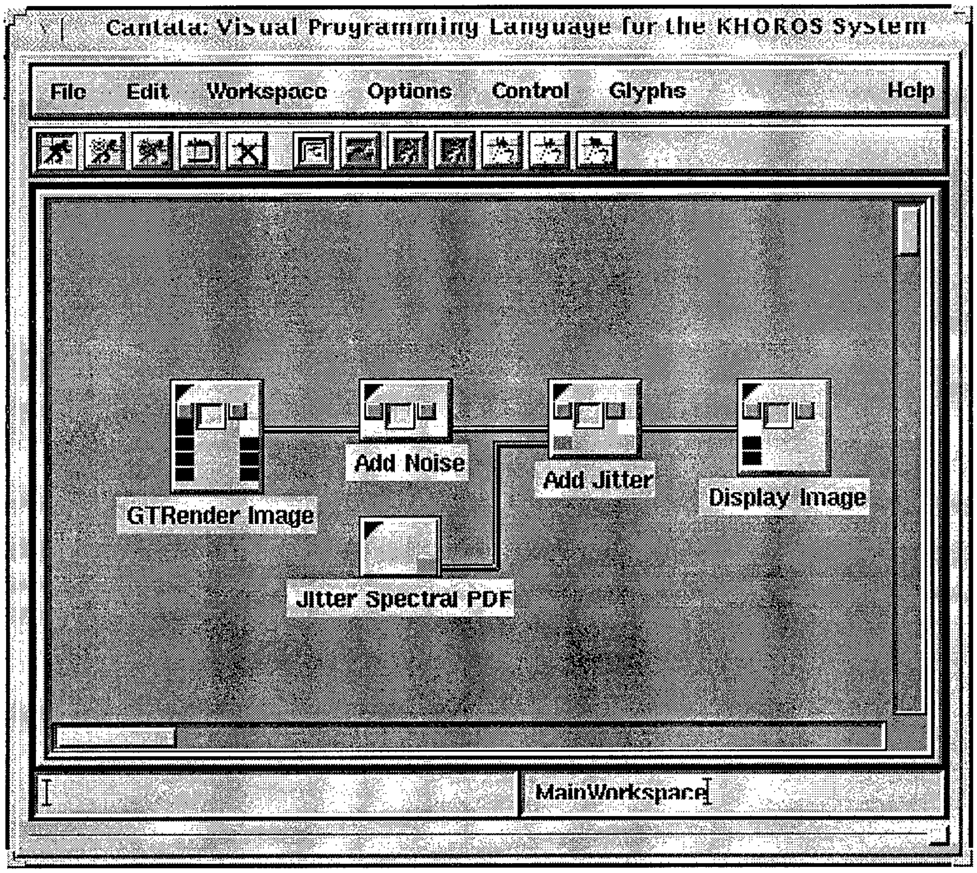
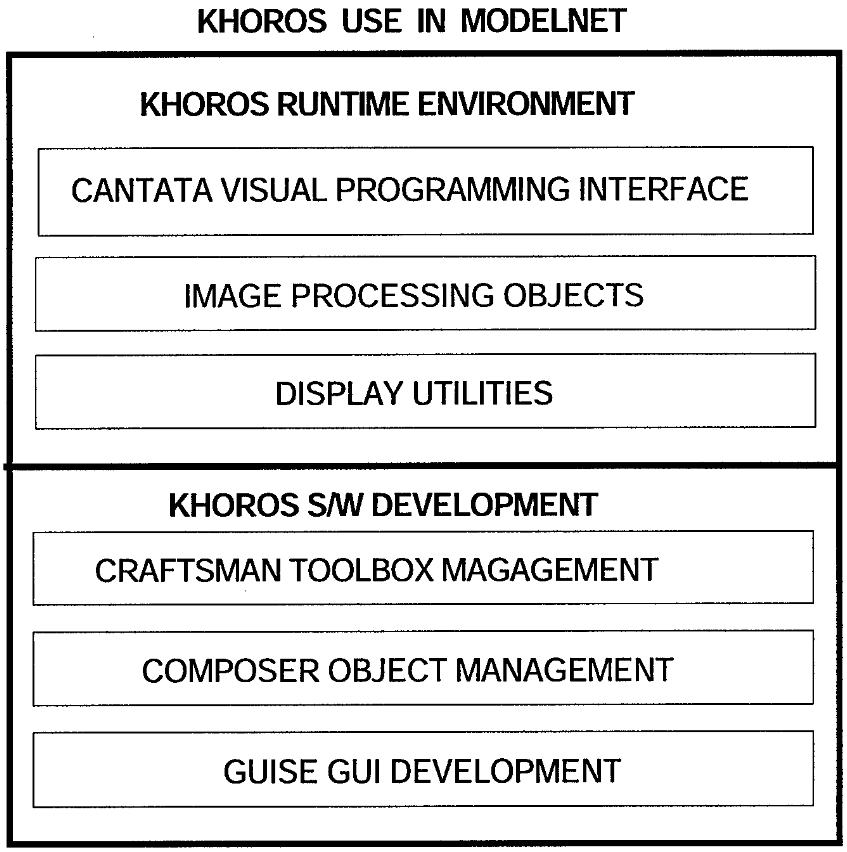


Figure 3 - Use of Khoros in the MODELNET environment.



KHOROS USE IN MODELNET

KHOROS RUNTIME ENVIRONMENT

CANTATA VISUAL PROGRAMMING INTERFACE

IMAGE PROCESSING OBJECTS

DISPLAY UTILITIES

KHOROS S/W DEVELOPMENT

CRAFTSMAN TOOLBOX MAGAGEMENT

COMPOSER OBJECT MANAGEMENT

GUISE GUI DEVELOPMENT

Figure 4 - Use of Browser in MODELNET.

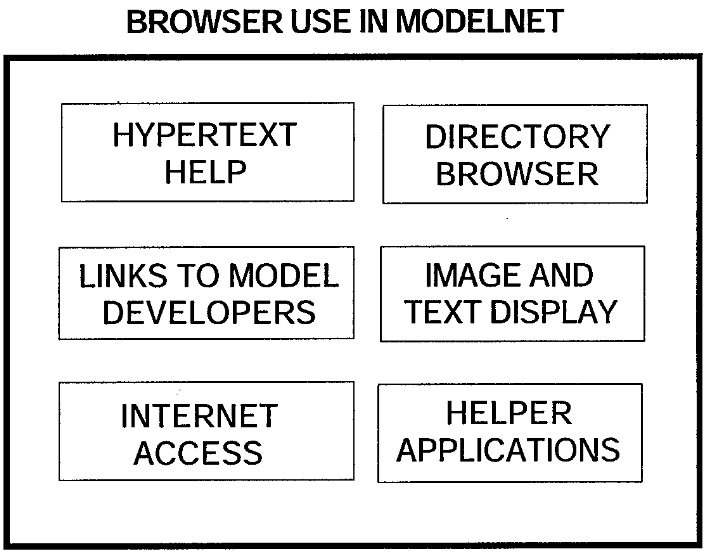
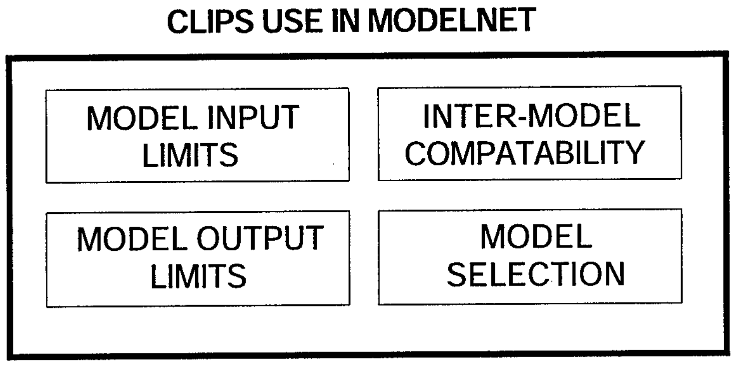


Figure 5 - Design of CLIPS use in MODELNET



Implementing a Blackboard Based Information Fusion Architecture Using the CEENSS Methodology

Presented by:

Mr. Harold W. Dean
Mr. Paul W. Salchak
SYMVIONICS, Inc.
1312 Research Park Drive
Dayton, Ohio 45432

Track No:
SIT2

Abstract

Two of the many problems confronting avionics developers today are the need to rapidly design, prototype, develop, and insert modern technology into avionics platforms and the need to support that technology under conditions where the pace of obsolescence is much smaller than the devices' required service life. SYMVIONICS' Complex System Information Fusion Tools (CSIFT) program is facing both of these issues. In this paper, we will present our findings and recommendations to-date as a case study of the goals, approach, and accomplishments of the program.

The CSIFT program uses a Data Activated Operating System (commonly referred to as a Distributed Blackboard), in the context of an offboard/onboard information fusion application, to implement a version of the Information Dispatcher function of the USAF's Airborne Information Fusion Architecture (AIFA). Our approach provides an efficient environment for prototyping and developing data oriented avionics applications.

The CSIFT program addresses the technology obsolescence issue by our use of the Continuous Electronics Enhancements Using Simulatable Specifications (CEENSS) methodology. This methodology provides for rigorous requirements modeling and formal verification, and promotes the use of vendor neutral, tool neutral simulatable specifications for electronic designs.

Background

A multitude of problems confront modern military system designers. Few of the problems are new. There are always funding constraints. Force reduction cycles are common occurrences at periodic intervals. There are labor issues, requirements changes, scope changes, and others too numerous to mention. For the most part, these issues have been faced by system designers for centuries. The first of the great pyramids in Egypt was not finished as a true pyramid because halfway through the project it was discovered that the initial design could not support it's own weight. There was an OOPS!

There is one issue facing modern system designers that has no apparent historical equivalent: the exponential rate at which information and technology are increasing and changing. More accurately, we all have been on an exponential curve. Current system designs are becoming very close to the section of the curve that is going vertical. Twenty years ago systems were planned for fifteen to twenty year life cycles. Today, at the completion of a three-year design cycle, up to half of the parts in system may be obsolete and unavailable. Software is not immune to the problem. The processor systems increase in capability, allowing new features, or become obsolete, requiring re-engineering.

Without attempting to belabor the obvious, it is essential to provide systems that facilitate fast, responsive, efficient hardware and software development and economical hardware and software technology enhancements and reengineering. SYMVIONICS' Complex System Information Fusion Tools (CSIFT) project with the Sensor Directorate is developing a flexible software development environment, the Blackboard, and providing a test case for a developing methodology for reengineering support, CEENSS.

In order to understand both the Blackboard and CEENSS, we must provide some additional information on the CSIFT project. The information provided about the CSIFT project and Sensor Fusion in this paper is presented to provide context for the Blackboard development tools and the CEENSS methodology. The sensor fusion technology being developed on the CSIFT project is a model application, the Information Dispatcher, to be the front end for a Sensor Data Fusion suite.

2 Sensor Data Fusion

What is Sensor Data Fusion? First, a caveat on this explanation. This discussion is not intended to be the authoritative history of Data Fusion, but to provide context, primarily with an Air Force bias. In general, Sensor Data Fusion initiatives are under the overall umbrella of Situation Awareness research and development. In modern air combat, pilots have more sources of data and information than ever before in history. In the early days of air combat, the pilots had some instruments and their eyeballs. However, technology marched on. Today pilots have the most sophisticated aircraft ever produced and they have more information than any person should need to sort through. They have voice data and AWACS data and satellite data and real beam radar and Synthetic Aperture Radar and digital maps and IR data and camera data and and and _ If it wasn't so important to actually fly the aircraft, accomplish the mission, and stay alive, it might be possible for the pilot to spend all his time sorting through data to find just the right piece. Situation awareness research is focussed on providing tools to assist the pilot with his understanding of his highly dynamic environment. Sensor Data Fusion, in general, focuses on the aspects of situation awareness that relate to on-board and off-board sensor data collection, correlation, and display.

Although all services are performing situation awareness and sensor data fusion, the CSIFT program traces it's roots to the Joint Directors of Laboratories (JDL) Data Fusion Group. The Joint Directors of Laboratories (JDL) Data Fusion Group produced the Data Fusion Process Model shown in Figure 1 as a reference for continued information fusion refinement and evolution. Significant efforts continue to improve the architecture, processor, and methods associated with information fusion for on-board and off-board avionics systems.

Figure 1. Historical Reference: The JDL Data Fusion Process Model.

A major impetus behind data fusion is the increasing numbers and effective range of weapons and sensor systems, resulting in an information war. The current international political climate will provide opportunities forever more fusion of this new and improved sensor gathered information. This information war and new political climate affects the strategy and tactics in the employment of systems, making data fusion a key and complex process, pervasive throughout all C3I systems. While the impact of data fusion is large, it must be recognized that fusion technology is implemented as a part of sensor, weapons, and C2 systems, and not as an independent entity.

Data fusion is fundamentally a process designed to enhance the value of information for decision support. It is an essential, enabling process to organize, combine and interpret information from various sources which may contain numbers of targets, conflicting reports, cluttered backgrounds, degrees of error, deception, and ambiguities about events or behaviors. A high level representation of the data fusion process, illustrating its major elements, is provided in Figure 1. Level 1 processing, Object Refinement, combines parametric data from multiple sensors to determine the position, kinematics, attributes or identity of low level entities. Level 2 processing, Situation Refinement, develops a description or interpretation of the current relationships among objects and events in the context of the operational environment. The result of this processing is a determination or refinement of the battle/operational situations. Level 3 processing, Threat Refinement, provides estimates of enemy capabilities and enemy intent, identifies threat opportunities, and levels of danger. Level 4 processing, Process Refinement, monitors and evaluates the ongoing fusion process to refine the process itself, and to guide the acquisition of data to achieve optimal results. These functions interact with each of the data function levels and with external systems for the operators to accomplish their purpose.

Based on the JDL model, the Air Force Research Laboratory (formerly Wright Laboratories) continued an effort to develop a generalized, open, non-proprietary avionics information fusion architecture. This architecture, the Avionics Information Fusion Architecture (AIFA), depicted in Figure 2, provided an architectural standard which defined the different functional areas, the underlying components/techniques for each functional area, and the high level interfaces between each component. This reference has since been advanced through various contracted efforts and the work of the Open

Architecture Fusion Work Group (OAFWG). For our purposes here, though, and in the interest of a simpler example we will use the AIFA for our remaining discussion.

Figure 2. Avionics Information Fusion Architecture (AIFA).

The CSIFT project promotes concepts and solutions that will support the evolving fusion architecture standards and standard elements. Specifically, we are focusing directly on on-board/off-board information problem, the Information Dispatcher, utilizing a "Blackboard" approach. The Blackboard approach has the following characteristics:

Σ Architecture is Extensible
Σ Architecture in Scaleable
Σ Architecture accommodate multiple processes
Σ Additional processors are easy to integrate
Σ Architecture is data driven (explicit scheduling of algorithms is not required)
Σ Integration is "plug & play"
Σ No explicit communication required
Σ Architecture is hardware independent

3 Blackboard Paradigm

What is a "Blackboard"? Well they used to be big sheets of dark colored slate that you hung on the wall and wrote on with pieces of chalk. Then they became green "something " hung on the wall, but you still used chalk. Now they are white and you use dry erase markers. "Blackboarding" was also an early decision technique in Artificial Intelligence applications that used a software "panel of experts" to evaluate a problem, each expert suggested a solution, then the decision heuristic combined the results to obtain "the answer".

The CSIFT Blackboard is not exactly any of the above but it makes use of some of the concepts displayed in the use of the above "Blackboards". The CSIFT Blackboard is a software operating system "extension" that provides a structure for developing and executing data driven software applications. We have based our work and initial software implementation on a Data Activated Operating System (DADS), developed by the Northrop Grumman Corporation. Extensions to the paradigm are being developed via our CSIFT project, with Northrop Grumman as a subcontractor.

3.1 Blackboard Paradigm

What is the Blackboard paradigm? Envision a large room with a traditional blackboard on one wall. The room has desks with "experts", all of whom have a small section of the blackboard that they watch diligently. Figure 3 illustrates the concept. Every so often, someone comes into the room and puts a sticky note with information (data), on the blackboard. The "expert" responsible for that section grabs the data, takes it to his desk, performs his function, writes new data on another sticky note, puts it on the blackboard in another section. At that point either another "expert" becomes active or someone grabs the note and takes it out of the room. None of the "experts" really need to know what anyone else does with the data. There is no explicit coordination of effort. Nobody really has to "schedule" each task. If a better "expert" becomes available to take over a desk, the process is not disrupted (as long the data remains in the same format). In the software implementation of this paradigm we rename the "experts" as Functional Elements and we identify the data as Data Elements.

Figure 3. Generalized Blackboard Representation.

It must be stressed that this simple concept does extend to multiple, distributed blackboards for more complex applications. Our current solution does not restrict their number or physical allocation and placement in a computing environment. In fact, as has been shared, one of the most exciting aspects of this approach is its natural extensibility and affinity for larger distributed configurations!

3.2 Data Fusion Blackboard

Now we can consider an example of one Data Fusion architecture - the AIFA (simplified to provide a manageable example) is depicted in Figure 4. We began with the following assumptions: A blackboard architecture exists which allows transparent distributed access to data. There are two essential components, the data objects, which are untyped, named byte arrays, and the functional elements, which consist of executable code which reads the data objects, as well as activation logic which determines when the functional element code is executed based on logical conditions which may exist in the data objects to which the functional element subscribes. The blackboard may be supplemented by intelligent agents, which in this context are dynamically created functional elements which are retrieved and instantiated from some persistent database. Functional elements do not care where a particular data object resides. Sufficient networking resources and abstraction in the API are assumed to make this possible. The basic philosophy is that the data objects capture all state' information, and are accessible to any functional elements to be read. They are written by exactly one functional element. All procedures or operations are represented by the functional elements. We have established the candidate blackboard allocations (Figure 5) described in the following paragraphs.

Figure 4. Simplified AIFA Example Architecture.

Figure 5. AIFA Example Blackboard Allocation.

3.2.1 Information Dispatching

This is the functional process responsible for all interaction with the outside world of data. The Information Dispatcher receives all data, both from on-board and off-board sources, reconciles temporal discrepancies, and routes data into the fusion structure. The Information Dispatcher also handles the data being handed back to the outside world.

3.2.2 Object Refinement (Level 1 data fusion)

The basic question answered by level 1 data fusion is "who's out there?" The operations associated with level 1 fusion are hierarchically decomposed as detection, tracking and identification. This suggests a partitioning of a blackboard dedicated to level 1 fusion into three sub-blackboards that are assigned to these three areas.

3.2.3 Detection Sub-blackboard

Functional elements assigned to particular sensors process reports, perform the appropriate coordinate transformations and post data objects to the detection sub-blackboard. As new reports come in the functional elements are free to prune the data as desired. For example, an FE may post the last three reports as separate data objects, or it might post the most current, replacing it as necessary.

Tracking Sub-blackboard

This sub-blackboard may be further decomposed into two sub-sub-blackboards: data association and alignment and track maintenance. Functional elements associated with data association scan the detection sub-blackboard and decide which data elements might belong together. At this point, hard or soft decisions may be made. In a hard decision, new data objects are immediately output which represent several aggregated detection reports. In a soft decision, a matrix might be output which represents pairwise scoring. An optimization algorithm (e.g. Munkres) could be used by another functional element to solve for the best possible pairings. The data objects involved in the pairing could be the detection reports only, or the detection reports plus the ongoing tracks (assignment of reports to tracks for updating tracks). The track maintenance sub-blackboard, reads the output of the data association (data objects with assignments to tracks) and updates the track data objects, which are then posted on it's blackboard. With the track maintenance sub-blackboard are functional elements associated with estimation. These would be used in two ways: to produce a synthesis of measurements from various sensors (for example a radar capable of range and azimuth only with measurement from a height finder radar), or to time align data (this is necessary in track filtering) by propagating state information. The estimation functional elements would not only be used by the track filters, but by any functional element requiring their functions.

Identification Sub-blackboard

The functional elements associated with this blackboard implement model based detection. They scan the track sub-blackboard. Each functional element is looking for a specific pattern with it's activation logic. For example, an FE might be looking for tracks that might correspond to fighter jets, based on platform dynamics. When a track object fits the particular pattern searched for, the ID-FE declares a hit and posts a data object to the identified track sub-blackboard. As in data association, at this point either hard decisions or soft decisions (possibilities which have some score) may be made at this point. This sub-blackboard represents the output of the level 1 data fusion blackboard.

3.2.4 Situation Assessment (Level II data fusion)

This consists of three identically structured blackboards: one each for enemy force refinement, neutral force refinement and friendly force refinement. The basic questions answered at this stage are "what are the groupings?" and "what do they seem to be doing?" The basic paradigm used here is model-based detection. Each functional element scans all of the identified track objects at the output of level 1 fusion and looks for a particular grouping. As such, each embodies a hypothesis to be tested such as "there is a squadron of planes flying in the same direction at location xyz". The geo-force refinement logic could be implemented as a sub-blackboard whose FE's get to score and vote on associations produced in the situation assessment blackboard, based on geographical properties, These could be instantiated dynamically as agents which correspond to what is possible for a given geographic region. For example, a situation assessment FE, reading a group of tracks, triggers and outputs a data object corresponding to a hypothesized tank formation at location xyz. An agent responsible for location xyz notes that the location corresponds to the middle of a swamp and that heavy vehicles such as tanks are unlikely. It, therefore, does not post an output data object or assigns a very low score.

3.2.5 Threat Assessment (Level 3 data fusion)

The threat assessment or level 3 fusion blackboard takes inputs from the level 2 or situation assessment blackboard. These consist of data objects that describe the friendly, neutral, and hostile force structures and activities. The purpose of the functional elements used by the treat assessment blackboard is to reason about the possible outcomes of these activities, based on weapons characteristics, rules of engagement and orders of battle. Probably the most realistic way of mapping this into a blackboard architecture is to have a controlling functional element corresponding to the conflict wargaming module which spawns agents as necessary to play out scenarios. The weapons models, tactics, rules of engagement, and strategy would be resident in a long term database accessible to and used to structure the creation of agents who play roles in simulations. The wargaming module would pose situations, create the participant agents, play out scenarios, and analyze the outcomes. The output would represent the likely outcomes and would give an estimate of threat intent. One possible advantage to approaching this as a simulation (set of simulations) is that tactics and weapons used by friendly forces may be evaluated as well and improvements suggested. The blackboard would be partitioned into three parts, an input section, with annotated objects from situation assessment, a scratch pad" section, used by the conflict wargaming module used to run scenarios, and an output section with likely outcomes and intents. Since the conflict wargaming module can execute evaluations in parallel, it would be able to answer "what if?" type queries posed by functional elements elsewhere in the system. One possible concern is the computational resources used to run simulations. There are several possible ways to address this. First, the hardware cost of computing is declining extremely rapidly. Faster processors and new techniques such as reconfigurable logic are making this so. Second, simulations may be run at many levels of fidelity and detail. Several functional elements corresponding to the conflict wargaming module could exist, to run the simulations at the appropriate level, or to use either heuristic or analytical models to save time.

3.2.6 Planning Object

The purpose of the planning object is to make recommendations that improve situation awareness. The most logical way to implement this in a blackboard architecture is to spawn agents which get to snoop the operations on the level 1,2 and 3 blackboards and post their findings on a blackboard associated with planning. Functional elements would then read these data objects containing synopses of the other blackboards and generate plans. As in other places, methods based on model based recognition could be used to look for specific situations and propose a plan. The planning object is one place where considerable effort could be expended. The nature of the operations here implies a goal matching mechanism that seeks to optimize use of resources to satisfy a particular user need.

At the output of the planning blackboard, a final set of FE's would score these plans and output a final set which would meet some threshold criterion. One advantage of doing this is that the set would be presented as options with tradeoffs.

3.2.7 Dynamic Integrated Situation Representation

This seems very straightforward to map to a blackboard architecture. The output data objects of the planning, aircraft status, threat intent and situation assessment blackboards would be read by an FE (or hierarchical set of FE's) that would determine what to display and how to display it.

3.3 Key Attributes for the Blackboard Method - a User's Perspective

We have just discussed how a representative portion of the Avionics Information Fusion Architecture (AIFA) could be implemented as a Blackboard system. Due to the nature of the Blackboard, any expert processes can be substituted to create any fusion architecture. Given this ability, then, a critical aspect concerns the benefits to be gained from such an implementation, in the context of the user of the IF system. In this sense, the user can assume several roles: human or machine, actively interacting with or merely monitoring the IF system. Furthermore, these roles may be set in the context of different activities, such as designing or developing the IF system, upgrading it, maintaining it, and dynamically interacting with it under mission conditions.

The strength of the Blackboard lies in its ability to serve the user in all these roles and within all these activities. The primary benefit is derived from the consequential ability to provide the user with a consistent view of the system from cradle to grave, in all phases of design development, testing and use under mission conditions. This allows the IF system implemented in the blackboard paradigm to be totally user-driven.

3.3.1 What Blackboarding Achieves for an IF Architecture

The whole notion of Blackboarding is to provide a data-driven viewpoint of a processing system, without disrupting the effectiveness or efficiencies to be gained through a structured or Object-Oriented (O-O) implementation of the actual processing software. Blackboarding extracts the functional-flow perspective from within, for example, the hierarchical and inheritance-driven structure of an O-O design. By extracting this perspective, as illustrated in Figure 6, and by making the key components of such a viewpoint (functional elements and data objects) accessible. Blackboarding provides three critical benefits to an IF system, or to any system for that matter:

Σ Interfacing Flexibility and State Saving
Σ User's context Processing, and
Σ Robustness

Although these components already exist in the O-O implementation of the system software and communications structures, their extraction into the Blackboard paradigm provides these three additional benefits, which we now discuss individually.

ÆINVALID_FIELD: ObjectØ

Figure 6. A general Blackboard, indicating key components.

Interfacing Flexibility and State Saving - These two capabilities are illustrated in Figure 7. As shown, the state saving is accomplished by the functional elements and the interfacing flexibility is inherent in the data objects. Although the functions and their states certainly already exist within the system O-O implementation, by assimilating several functions to post data to a Blackboard, we capture the relational context of these functions, their states and the particular data as some functionality of interest to a user, which we will then make available to that user through an interface. In other words, this aggregation of state, function and data has relevance to a using system, intelligent agent, or human, so we make it available easily for interfacing and understanding of the relational context that matters, in a much more direct fashion than instrumenting the O-O structure to capture such information.

ÆINVALID_FIELD: ObjectØ

Figure 7. Capturing function, state, and data relational relevance in a Blackboard.

The myriad relational relevances of sets of function, state and data, are the very heart of an IF system. Hence, the ability to capture any number of these relational views is exactly one-third of the key to realizing a truly user-driven IF system under all the roles and activities we defined earlier. The second one-third has to do with maintaining the user's context, as we discuss next.

Processing in the User's Context - In an IF system, as well as many other complex systems, the user's view is not necessarily that of the software designer or system designer. It is, however, the user's view which should drive design, implementation or modification decisions. Obviously it is the user's view which applies to the use of the system.

Requiring the user to penetrate or understand the nuances of the software design or the computing system design is unreasonable. Especially in IF, where the fusion processing is complex in itself, we should not require the user to place their view of the functional flow and process sequencing in the framework of the software or system design. Whether the user is human or machine, active or passive, their view of the system should be the only view they need to deal with. The Blackboard technique provides such a view.

Consider the simplification of the blackboard view of some arbitrary functional elements and data objects of Figure 8. Here, we have overlayed the basic functional view of this particular system segment over the shadowed data object and function structure. This is precisely the view the user requires, with emphasis on functional flow, visibility of data, and access to functional state. This view is relevant to the user, without the encumbrance of the software or system representations. In the context of this view, the user may view, capture, or interface the IF system within the data-driven and functionally-concise view of importance.

Figure 8. Capturing the concise functional view through the Blackboard.

Now, having the ability to interface easily, maintain state, and view relevant portions of the IF system in the user's context, we have two-thirds of what we need for an effective, user-driven IF system under all the roles and activities we defined earlier. The final one-third has to do with robustness, as we discuss next.

Robustness - In the IF context, robustness has to do with improving design or development effectiveness, easing future modifications and upgrades, and custom-tailoring the IF system to a particular mission need. In all of these areas, the IF system needs to permit the addition, deletion, or re-organization of functional elements and the data objects relevant to the activity in a simple and intuitive manner. As illustrated in Figure 9, the Blackboard system permits such ease of change.

ÆINVALID_FIELD: ObjectØ

Figure 9. Robustness of the Blackboard system.

Here, we note that adding (or deleting) functional elements, rearranging connectivity, or posting additional data objects to the Blackboard may all be done easily and in the user's context. Whether for design, testing, operational optimization, or any other reason, the changes can be done purely in the user's functional context. Hence, the user's needs, whether for human or machine interface, interactivity, or simple monitoring, are always clear in the view of the system which is to satisfy these needs.

In summary of the Blackboard's ability to support the general task of IF system implementation and application, the ease of interface, state saving, view in the user's context, and robustness make this technique especially beneficial. Fundamentally, Blackboarding extracts the appropriate view or views, whatever they may be, for the user, whatever or whoever that may be, in the user's particular activity. As Figure 10 shows, it extracts the concise, relevant process view from the "clutter" of the (typically) O-O software system executing upon some processing system or systems. Optimal software system design is not being trivialized here, and the benefits to the overall quality of a software design from O-O's hierarchical, modular, inheritance-driven paradigm are valid and necessary. However, these issues are the domain of the software designer, not the IF user. The Blackboarding technique permits both systems to co-exist efficiently, and provides a means for the user to extract the proper view of the system functions and their important relational aspects.

ÆINVALID_FIELD: ObjectØ

Figure 10. O-O and Blackboarding - an effective separation of view.

3.4 Blackboarding as a Design Aid

Consider the design representations of Figure 11. Here, we see the division of viewpoints for the different design disciplines. The Application Designer, in the IF situation, is mission-conscious and process-oriented. This designer seeks to organize some arrangement of processes to refine source information such that information critical to the mission is most effectively handled in a decision support role. The Application Designer is conscious of the need for efficient and cost-effective software and system design, but is usually more in the role of setting requirements for such endeavors than actually considering them. The Software Designer lives in the world of modularity, reusability, and O-O design methods by which these, and other cost-effectiveness goals, may be accomplished. The process and data flow of the Application Designer exists in this realm, but the Software Designer concentrates more upon vertically integrating processes, through nested methods and inheritance aspects under the O-O paradigm. The System Designer works in the realm of databusses, processors, and physical constraints. This designer must support the requirements of the Software Designer through effective organization of processing and storage elements and their communications links. The process and data flow of the Application Designer is remote to this area, where the requirements from the Software Designer drive the system implementation.

ÆINVALID_FIELD: ObjectØ

Figure 11. Multiple perspectives of the design process.

Since the Application Designer needs an understanding of the process and data flow as the design is producing, but not necessarily the O-O structure of system configuration, the Blackboard provides an ideal means to separate and present that view. The Blackboarding can capture the relevant processes and data objects to an aspect of interest to the Application Designer, who can view this process and data flow perspective as the design matures. All the Application Designer requires is a concurrent development of the Blackboarding elements (Blackboard, interfaces) with the software and system development. Since software development is generally conducted in a spiral fashion, with iterative and progressively complete implementations at all levels of the design, concurrent implementations of the Blackboard and its interfaces will keep the Application Designer focused precisely upon the relevant aspects of the design to satisfying the IF requirements.

From this example, we can draw some conclusions. The Blackboard interfacing must do only one thing: configure the design data for its intended application as a design visualization tool. Unlike the dynamic application situation, there is no need for any kind of "state" information, as the state of the design IS the design. We can extrapolate this simple example to the other aspects of life-cycle as well. If the Blackboard can interface to the supplier system in any of its manifestations, design, prototype, multi-iteration versions, pre-production, and final, then the Blackboarding system can be the Application Designer's tool universally, from cradle to grave. In the test manifestations, state-saving once again becomes relevant. Basically, we can use the Blackboarding technique as the designer's viewer, subject to the same two requirements we defined in testing the flexible interface and state-saving hypothesis:

Σ The input and output interfaces to and from the blackboard must be possible, and

Σ State information must be provided along with control or information data.

We can conclude from this simple example and its extrapolation to the more complete life-cycle, that our hypothesis regarding the ability to provide a consistent user's view of the AIFA is valid, and that such a view has value to all aspects of an IF system's life-cycle, subject to the requirements we asserted. Again we defer more comprehensive discussions of the requirements to the conclusion of this section, and continue into the application aid case.

3.5 Blackboarding as an Application Aid

To optimize some facet or facets of an IF engine, visualization or AI-processing systems (intelligent agents) need to capture process and data information precisely into their particular user's view of the system and data. Our studies revealed that the user, which could be an intelligent agent (machine or human), could test hypotheses based upon the data presented through the blackboard, to optimize the IF system via control functions. For example, assume a mission manager function in the AIFA architecture, which includes an intelligent agent whose purpose is to optimize the probability of optimal sensor mixing.

Here, similar to the case we considered under the flexible interfacing and state-saving proof, our Blackboard includes a posting interface and some output interfaces. The Blackboard captures several information and control data objects, including:

Σ The Resource Manager commands to the Information Sources which reveals how these resources are being commanded,

Σ The information output of these sources, which reveals the results of executing the commands,

Σ The information provided to the Enemy Force Refinement from the Information Dispatching, which reveals the information upon which it is acting, and the result of the Threat or Target Intent processing, which is the conclusion of this processing sequence.

In other words, the Blackboard captures what we are commanding, what information we are getting from the sources, and what our intent processing is deciding based upon such information. We may then provide some or all of this information, with its associated state, in whatever format we choose to the human user, by interfacing it to the PVI interface function. We may also provide this information, via a system interface, to some intelligent agent operating under the Mission Manager to optimize this sub-processing sequence within the AIFA. Hence, we have the ability to form the user's view from the AIFA, in a particular context for a particular reason, and could easily do so for myriad other context and reasons, subject to the same requirements we have seen for the previous tests:

Σ The input and output interfaces to and from the Blackboard must be possible, and

Σ State information must be provided along with control or information data.

We can conclude from this example and its extrapolation to the more general case, that our hypothesis regarding the ability to provide a user's view as a dynamic application aid is valid, subject to the requirements we asserted. An once again we defer our discussions of the specifics of these requirements to the conclusion of this section. Next, we look into the robustness issues.

3.6 Implementing Future Modifications

The Blackboard approach is essentially "made to be modified." As illustrated in Figure 12, upgraded or new processing entities can be integrated with a process and function view (application designer), as well as the necessary software and system implementation views (software/system designers). Any new processes and their associated data appear in the process and data flow view to the Application Designer, as methods and data in the inheritance of the O-O structure to the Software Designer, and as processing, storage, and communications loads to the system designer.

Another aspect of robustness which our investigations revealed is that the consistency in user's view provided by Blackboarding can not only extract the proper perspective in the design stages, but can actually enhance the whole process. The Blackboard perspective remains consistent across all portions of a software designer's iterative development paradigm: design, implementation, testing, assessment, revision, and so forth. This consistency of view ensures that the Application designer is responding to issues raised in the various iterative development stages with the proper understanding.

ÆINVALID_FIELD: ObjectØ

Figure 12. Upgrading with the multiple-perspective views.

3.6.1 Providing Customized, Interactive User Interfaces

This capability can build directly on top of the blackboard context, again leaving the specific implementation of the software and hardware system functionality to their respective domain specialists. We basically proved this capability in the context of the previously discussed Intelligent Agent addition to the AIFA. The only additional issue is if we wish to implement such customized interfaces in-mission. Then we must determine whether the user's view of the system can actually be reconfigured in a manner that it does not disrupt ongoing IF functions critical to mission performance. The issue is not whether or not it can be done, but how to synchronize such an action into the dynamic mission execution. Hence, we see an additional requirement: the customized interfaces, if they are to be dynamically reconfigured in-mission, must not disrupt ongoing IF or other mission processing functions.

Basically, ignoring the in-mission aspects, the ability to customize the IF system for the user through the Blackboarding technique can be thought of as a layered system, with each layer in a particular specialization domain, and the focus of the whole implementation is upon consumer (user)-driven IF system optimization to satisfy diverse and changing mission needs. This is illustrated in Figure 13. As we disclosed earlier, the entire blackboard concept is also made to be modified, since the same consumer-driven paradigm for application also holds true for design, implementation and modification. By providing the user a consistent view of the system in all phases - design, implementation, modification, and application, we ensure a complete, user-driven system. Since it is an open architecture, data-driven, and object-oriented, ease of enhancement is built-in right from the beginning.

ÆINVALID_FIELD: ObjectØ

Figure 13. A layered perspective of the Blackboard as applied to IF.

3.7 Blackboarding and an Information Fusion Testbed

We have discussed three principal features of Blackboarding which demonstrate its support within an Information Fusion architecture. These same three attributes also permit the Blackboarding technique to serve as a common integration framework for the purpose of implementing a Distributed IF Testbed - which can serve all developers, testers, and users of IF technology in their particular missions and endeavors. In this section, we describe how this support for an Information Fusion Testbed can be achieved.

We begin with an abstraction of the view of Blackboarding for IF we presented at the conclusion of the previous section. Basically, this is a layered view of the Blackboard and its interfaces implemented upon some set of applications executing upon some array of processing systems. The interfaces basically provide users, or consumers, in-context access to the application/system sets, or suppliers. If we wish to make such a structure compatible with many different consumers or suppliers, we need only to implement the appropriate interfaces, as illustrated in Figure 14.

ÆINVALID_FIELD: ObjectØ

Figure 14. Multiple consumer/supplier interaction via the Blackboard.

The general strategy for the evolution and optimization of IF technology is to continue developing complementary Air-to-Air (A-A) and Air-to Ground (A-G) IF technology, by implementing on the ground first, and then transitioning these technologies to the air. An critical aspect of this paradigm is that it requires the development of a testbed, that is, an infrastructure and technology base to permit the development and testing with maximal use of real-world data and cooperation and collaboration of world-class talent. A distributed IF Testbed is the framework by which such talent, data, and technology may be organized collaboratively to develop the IF we need for the next century. The Blackboarding technique can provide that framework, through its ability to interface myriad applications, systems, and users, and to provide these users their necessary viewpoint and context. We will demonstrate this concept by beginning with a snapshot of a real-time embedded IF architecture, set in the Blackboarding framework. Then we will show how this system's development, testing, and implementation may be achieved under a Blackboard-supported testbed concept.

Consider a Fusion Manager function, as illustrated in Figure 15. Typically, and consistent with the AIFA or JDL fusion paradigms, such a function would be engaged in gathering information from source objects such as sensors and CNI systems, and invoking stages of object and threat refinement to provide a threat or target assessment to the Mission Manager. In this subsystem of the overall IF process, the consumer is the Fusion Manager, which receives information in a Fusion Manager's view of the situation, through an interface to the Blackboard. Information from the sources each have their own interfaces to post data (information and state) to the Blackboard. in an active mission, the Fusion Manager would also direct a sensor manager to reconfigure the source (sensor) mix and modes, although we do not show that aspect in the figure.

ÆINVALID_FIELD: ObjectØ

Figure 15. Blackboard supporting a Fusion Manager - real-time application.

Now, consider how this Fusion Manager might have been developed and tested, using Avionics Laboratory Assets. Assume that the MBV (Model-Based Vision) Laboratory had some sensor models which would serve well as test benches for the sensors, and that JMASS included environment and CNI models which would provide realistic simulation of these aspects. An IF Testbed would need to integrate these two resource laboratories (MBV and JMASS), and maintain the Fusion Managers view, to permit development and testing.

Figure 16 shows how the Blackboarding can support this activity. Here, the Blackboard provides the means to permit the KHOROS-framed MBV Lab sensor models and tools to interact with the JMASS-framed environment and CNI models. The Blackboard also extracts the Fusion Manager's viewpoint, and can support multiple subscribers (consumers) in a situation where different individuals or organizations may be responsible for, or interested in, different aspects of the Fusion Manager design, development or testing. The MBV Lab is not concerned about the fact that they operate in a KHOROS framework upon Unix systems, and that JMASS operates in a Solaris environment. The JMASS facility is similarly unconcerned about inter Lab particulars. The developers, designers, or testers (subscribers) viewing the simulation results are unconcerned with any of these inter-lab application or system issues, and concentrate on the function and flow in the context of their needs. The Blackboard enables all of this.

ÆINVALID_FIELD: ObjectØ

Figure 16. Blackboarding managing lab assets for subscribers in a development or testing application.

Now we can extend this concept to a distributed situation. Let us now assume that we are using not only Avionics Laboratory assets, but contractor's or other government facility assets at geographically-dispersed facilities. We might also consider that the development or testing is a collaborative effort, involving users (consumers) at geographically-dispersed locations. Implementation of a Distributed IF Testbed, under the Blackboarding paradigm, is again merely a matter of proper interfacing, as shown in Figure 17.

In this situation our interfacing has two classes - local and remote. Here, a CORBA (Common Object Request Broker Access) implementation would serve well as the local interfacing paradigm. Our figure is of course a simplified view, as CORBA shells and other CORBA structures would permeate into the applications in a fashion more than a simple interface "block." A Java-based remote interfacing construct would serve the remote users very well, as the Java applet and byte coding capabilities would provide browser-based consumer-supplier interaction as well as visualization of the testbed operation and execution. In fact, a Java-based, "Intranet" implementation, using CORBA at the intermediate levels, might be a good, common access paradigm for the IF Testbed in general.

ÆINVALID_FIELD: ObjectØ

Figure 17. Blackboarding to enable local and remote access to a geographically-distributed IF Testbed.

4 A Word About CEENSS

The Continuous Electronics ENhancements using Simulatable Specifications (CEENSS) program seeks to establish methods and tools for stable, repeatable, manufacturable electronics designs by focusing on formal requirements modeling, requirements verification through mathematical proof, and a complete, unambiguous, executable specification - called the SimSpec. The overall concept, depicted in Figure 18A, has shown outstanding improvements in overall design and design maintenance time - especially over a life-cycle of change, reducing risk and cost-to-market. A composition of a SimSpec is shown in Figure 18B. The CEENSS approach also centers on the use of 'Design Shelving' as a critical aspect of its Product Development Process. This framework is summarized in Figure 18C.

Figure 18. The CEENSS Concept.

Figure 18B. The CEENSS SimSpec.

Figure 18C. The Design Shelving Framework.

SYMVIONICS has successfully applied the CEENSS Spiral Development Methodology and many of the projects' tools during the CSIFT program. These "lessons learned" have been shared through involvement in the CEENSS Industry Review Board (IRB). For more information about the CEENSS program, and its current status and progress, refer to the web-site at http://www.ececs.uc.edu/~kbse/ceenss/.

5 Summary

In conclusion, the CSIFT Blackboard represents an exciting step in high performance distributed operating environments, with potential applications well beyond its information fusion roots. We have seen outstanding characteristics in both development (rapid, expressive, extensible, partitionable) and execution (behavior, performance) references. Our work will result in a set of open architecture tools and hardware demonstrating the initial off-board/on-board fusion information dispatcher, but also suitable for experimentation and application to other domains. We welcome any opportunity for discussion, critical analysis, additional research, or transition into developmental applications. Contact either author at (937) 426-4504, or through psalchak@symvionics.com.

Biographies:

Mr. Harold W. Dean

Mr. Dean is currently the Director of Programs for Research & Technology for SYMVIONICS, Inc. He is a graduate of The Ohio State University with a Bachelors degree in Electrical Engineering. He has 20 years experience in the design of military electronics systems. His primary expertise is embedded processors, aircrew training simulators, and high speed data acquisition and distribution systems. He is the co-inventor of the "Method for Simulating High Resolution Synthetic Aperture Radar Imagery From High Altitude Photographs," Patent Number 5,353,030 issued 4 October 1994. He has previously published the paper "Network Switching in the Virtual Test Station (VTS)" which was presented at the Test Facility Working Group Conference (TFWGCON), 1995.

Paul W. Salchak

Mr. Salchak is currently sharing responsibilities as a Staff Scientist and Director of New Products & Programs for SYMVIONICS, Inc. He is a graduate of Rochester Institute of Technology with a Bachelors degree in Electrical Engineering, and has completed a variety of graduate and specialized studies nation-wide. With over 19 years of experience in high-performance human interactive systems, Mr. Salchak is an expert in system analysis and development, electronic systems design automation, simulation-based development methods and tools, processes and metrics, training systems, simulation technologies, high-performance computing environments, and avionics systems, sensors, and displays. He is the inventor of the rapid-concept-to-market technique, Assertive Development, used by SYMVIONICS' research programs to define, insert, qualify, and transition advanced technologies. Recent research topics and reports have spanned such diverse areas as information distribution and leveling, chaotic signal processes and discrimination, collaborative/distributed decision support techniques, hardware/software co-design, Diminishing Manufacturing Sources (DMS), and Distributed Mission Training (DMT). Mr. Salchak's most recent publications have included "Supporting Hardware Trade Analysis and Cost Estimation Using Design Complexity" (presented at the VIUF National Convention/Fall '97) and "Innovative ATR Through Statistical Signal Processing and Chaos" (presented at the recent GOMAC '98).

Software Factory 2001

S. Wayne Sherer, 0.5. Army TACOM LCSEC Jack Cooper, Anchor Software Management

Introduction

Industry has long recognized the advantages of the product-line approach to development. This paper synthesizes several Department of Defense (DoD) software-related initiatives into the Software Factory 2001 concept, describes the software factory process flows, and presents some of the challenges for implementation.

A long series of initiatives illustrates DoD's desire to become more product-line oriented. Long-standing initiatives encourage reuse and reengineering, while other initiatives include the program executive officer (P50) system, the common operating environments (COF), greater use of commercial-off-the-shelf (COTS) software, and open systems. New technological tools have become available that facilitate some of these initiatives, such as domain modeling for defining product-line architectures, and Capability Maturity Models (CMMs) to guide process improvement. The Software Engineering Institute (SEI) recognizes the need for product-lines in draft Version 2.0 of its CMM for Software. This version includes a new Level 4 key process area called Organization Product Alignment, which states that organizations at this maturity level establish and maintain product-lines and associated assets.

Many DoD software organizations find it nearly impossible to deal with all of the above initiatives at the same time. Yet most of these initiatives were based on best practices that have found wide commercial success. Perhaps these initiatives are difficult to implement in DoD organizations because they were adopted from successful commercial organizations that operated under an entirely different development paradigm, the product-line approach. Another reason maybe that these initiatives were established one by one over time and are addressed individually by DoD instead of as a group. Software Factory 2001 draws on and combines all the above initiatives into a unified approach, while incorporating them with the successful industry-based product-line approach that inspired these initiatives.

What Is the Software Factory?

In contrast to DoD's past creation of multiple stovepipe, stand-alone systems, the software factory for the year 2001 (Factory 2001) draws on the commercial industry's success with creating product-lines or families of related products. The product-line approach tends to be more evolutionary than revolutionary; system architectures evolve with new technology and requirements rather than being reinvented for each system development. Newer members of the product-line base their developments on previous product versions, and they reuse processes, architecture, design, tests, and other artifacts.

Software Factory 2001 institutionalizes the product-line concept and uses it as the skeleton for it's engineering process. Like commercial companies, DoD organizations specialize in certain products, such as tanks, radar, compilers, or tools. DoD organizations can use the Factory approach as a structure to evolve into a commercial model. Over time, the organization's culture and policies acclimate to the product-line approach to building products as it builds a legacy (unwritten history of the process) and develops collective expertise and a reputation for it's product-line areas.

To achieve efficiencies similar to industry, the DoD must reorganize system development and lifecycle support around product-lines. This reorganization is the first step toward institutionalizing Software Factory 2001. It will be difficult to move from a stovepipe orientation to product-lines, but well worth the aggravation in terms of improved efficiency, product quality, and compatibility.

A primary obstacle to this method of development is that DoD currently funds systems instead of product-lines. Fortunately, DoD already has some de facto product-lines such as fighter aircraft, submarines, tanks, logistics systems, and trainers. The PED system also defines some product-lines. The most difficult challenge is the change in the culture that must take place to overcome the inertia developed from the many years of stovepipe funding, system development, and lifecycle support.

Software Factory 2001 Environment

The environment for Software Factory 2001 is largely different from today's, it is tailored to the domain of the product-line and is standard across all software-engineering efforts in the factory.

DoD "Building Codes"

While most industrial domains, such as automobiles, plumbing, telephones, and farm machinery, have had standards in place for years, the software community has few widely accepted standards. A set of software building codes (or standards) underlies the Software Factory 2001 common operating environment. These codes serve the software purpose in the same way building codes serve the construction industry. Instead of 2 by 4s positioned with 16 inches between centers, the software building codes are standard data definitions, interface standards and message formats. These building codes should include open system and interoperability requirements necessary to ensure that assets within a product-line are portable and can exchange any needed information.

The Army has adopted a set of software building codes with its technical architecture and is in the process of implementing them. The Office of the Secretary of Defense has adopted and is maturing the Joint Technical Architecture (JTA) for all of DoD. These software-building codes must be in place and implemented for Software Factory 2001 to be a reality.

Common Operating Environment

Another fundamental element of Software Factory 2001's environment is the DoD's formation of a COE. The COE is the source of ingredients, tools and processes from which Software Factory 2001 builds its software systems. The repository contains the assets from the product-line's legacy system architectures, requirements, design, code, algorithms, and test cases that have been certified for reuse. Another repository component is the COTS software that is applicable to the product-line. Also, technical architecture building codes and cross-domain reusable assets from other COEs, such as Ada run-time kernels and kalman filters, are included.

The COE contains the factory's software engineering environment (SEE). The SEE is the repository of tools to build software systems. The COE contains the definitions and training materials for each component of the standard software engineering process. Other items include middleware products that can isolate applications from the underlying hardware and operating system platforms and thereby improve portability When a new system is to be composed, the COP is entered and the appropriate ingredients selected. The COP concept is catching on within the DoD and will expedite the enabling of Software Factory 2001. The critical factor is that the COP is all-inclusive and that reuse includes all COE elements not just code.

The Software Factory Process

A central part of Software Factory 2001's operating environment is its standard software engineering process, which must be overseen by managers committed to managing it. A standard software engineering process is important: it helps eliminate communication and compatibility problems, its outputs are predictable, and it lowers risks. Organizations at maturity Level 3 in the CNN for Software inherently have standard processes. Figure 1 shows the top-level sequence of the software factory process and is the basis for the rest of this section.

Figure 1 Software Factory 2001 Process
When the product-line has not yet been established (see Figure 1 dark and medium gray chains)

Managers must select a representative set of systems for the domain, Conduct a domain analysis of these systems and of customer needs and expectations to understand the commonality and variability across the domain Establish a Domain COE based on the JTA and compatible assets from other COEs Develop a product-line architecture that leverages the commonality, bounds the variability to the extent feasible and allows for the unique variability of each family member
Enter certified assets, based on the product-line architecture, into the COE, sources include:
Assets engineered from the product-line architecture
Assets from existing systems reengineered to the product-line architecture Compatible legacy assets from the representative systems

This initial COE maybe minimal and it must be followed by an enrichment effort that engineers or reengineers additional items that will be compatible with the new product-line. The assets selected for engineering or reengineering should be selected based on their cost and value to the product-line. Mote that enrichment of the COE can also be accomplished incrementally as a by-product of building new members of the product-line. Each asset in the CDE must be certified via evaluation to establish it's reliability, in the context of the product-line, so future users will have faith in the asset or know the risks involved in using the asset.

With a product-line architecture and the initial CDE in place, the factory is ready to build systems (see Figure 1 light gray chain)

When a Software Factory 2001 organization receives an order for a new system, it must decide how to build the new system. There are four major categories, each of which requires a different instantiation of the factory process:

Build a new system according to all the rules of the current product-line.
Build a new system that requires new functionality not in the current product-line
Reengineer an existing product-line-conforming legacy system.
Reengineer a nonconforming legacy system.

Each instance of the process begins with analyzing customer needs and expectations.

A. Building a New System Using the Current Product-line Architecture

Step 1

Customer needs and expectations are analyzed to generate requirements.
The requirements for the new system are elaborated within the constraints of the product-line architecture.
From the beginning, the new system architecture conforms to the standard product-line architecture.
Standard building codes provide the underlying framework to define how the new system is to be built
Requirements elaboration continues cyclically until the requirements stabilize; elaboration can also include prototypes composed from assets in the COE. The result is an architecture for the new system that is a subset of the current Product-line Architecture.

Step 2

Once the requirements are stable, any prototype(s) is discarded and composition of the new system occurs within the constraints of the standard product-line architecture.
The new system's architecture is used to navigate the COE and select the assets from which to compose the new system, e.g., tools, processes, COTS, reusable software, middleware, generators, and test cases.
Most of the time the composition will result in a partial system that requires additional engineering.
In the course of engineering the new system, it may be determined that some of the new system assets are candidates for incorporation into the COE. These assets are packaged appropriately, and once certified for reuse and for compatibility with the product-line, they are added to the COE.
In some cases system composition can completely build the system from existing assets.
With engineering finished the system is ready for certification.

Step 3

The system is certified against the requirements to ensure customer needs and expectations are satisfied.

B. Building a New System that Requires New Functionality

Step 1

Customer needs and expectations are analyzed to generate requirements.
The requirements for the new system are elaborated within the constraints of the product-line architecture to the extent feasible.
Standard building codes provide the underlying framework to define how the new system I to be built; however the new functionality may require additional building codes.
During requirements elaboration the new functionality must be analyzed to determine if it will be required in future members of the product-line If yes, the standard product-line architecture is updated to provide for and allow the new functionality.

If not, tradeoffs between the system architecture and the standard product-line architecture are made that best accommodate the new functionality Requirements elaboration continues cyclically until the requirements stabilize; elaboration can also include prototypes composed from assets in the COE and may include additional updates to or tradeoffs with the standard product-line architecture.
The result is an architecture for the new system.

Step 2

Once the requirements are stable, any prototype(s) is discarded and composition of the new system occurs as appropriate within the constraints of the standard product-line architecture. The new system's architecture is used to navigate the COE and select existing assets that can be used in the new system, e.g., tools, processes, COTS, reusable software, middleware, generators, and test cases.
Management may decide to start a parallel effort to reengineer assets from existing systems to satisfy part of the new functionally in the system. These actions will result in a partial system that requires additional engineering.
In the course of engineering the new system, it may be determined that some of the new system assets are candidates for incorporation into the COE.
When the new system's requirements were incorporated into the standard product-line architecture, enhancements or additions to the COE should result. These assets are packaged appropriately, and once certified for reuse and for compatibility with the product-line, they are added to the COE.
When engineering is finished the system is ready for certification.

Step 3

The system is certified against it's requirements to ensure customer needs and expectations are satisfied.

C. Reengineering an Existing Product-Line-Conforming Legacy System

Step 1

Customer needs and expectations are analyzed to generate requirements. Candidate legacy systems are analyzed as part of requirements elaboration. Once it has been determined that a legacy system is a suitable starting point to build the new system, the new requirements are elaborated while conforming to all of the constraints of the product-line.
From this point Step 1 is the same as building a new system using the current product-line architecture.

Step 2

Same as building a new system using the current product-line architecture, except that the legacy system is treated as if it is stored in the COE.

Step 3

Same as building a new system using the current product-line architecture.

D. Reengineering a Nonconforming Legacy System

Step 1

Customer needs and expectations are analyzed to generate requirements. As part of the cyclic requirements elaboration the nonconforming legacy system is analyzed for its compatibilities and incompatibilities with the product-line. If it is determined that the legacy system is a suitable starting point to build the new system, the requirements are elaborated within the constraints of the existing system architecture, accommodating the product-line architecture whenever possible.

The constraint of the existing architecture limits the progress that can be made toward achieving a new system fully compatible with the product-line.

From this point Step 1 is the same as building a new system that requires new functionality.

Step 2

Once the requirements are stable, any prototype(s) is discarded and composition of the new system occurs as appropriate within the constraints of the standard product-line architecture.

The legacy system's architecture is used to navigate the COE and select existing assets that can be used in the new system, e.g., tools, processes, COTS, reusable software, and test cases.

The legacy system is reengineered to satisfy the requirements for the new system using Software Factory 2001's standard software engineering process.

Design, development, and composition proceed within the constraints of the existing implementation.

In the course of engineering the new system, it may be determined that some of the new system assets are candidates for incorporation into the COE.

When the new system's requirements were incorporated into the standard product-line architecture, enhancements or additions to the COE should result. These assets are packaged appropriately, and once certified for reuse and for compatibility with the product-line, they are added to the COE.

When engineering is finished the system is ready for certification.

Step 3

System requirements certification is performed using the standard engineering method.

Reality by Example

Software Factory 2001 is not an academic exercise; all of the pieces are available today. To demonstrate, an existing Army Life Cycle Software Engineering Center (LCSEC) will be used to show a specific implementation of most of the pieces of Software Factory 2001.

Mapping the Specifics

The following maps a specific Army implementation onto the above Software Factory 2001 processes.

Facility - The Tank-automotive and Armament Command (TACON) LCSEC, Picatinny Arsenal, NJ. This is one of three Army organizations that provides lifecycle software support to battlefield automated systems. It currently supports over 55 systems.

Product-line or Family - Fire control systems for tanks and self-propelled howitzers (cannon) are the LCSEC's main product-line; however, it also has
other product-lines related to biological and chemical detectors, intelligent armaments, and gunnery simulators and trainers.

Infrastructure - An established Army software support activity for over 15 years, complete with personnel, resources, training, contractor support, collective domain expertise, research, and an organizational policy.

Capital Equipment - At the LCSEC, the DOE is populated with legacy software and documentation for over 40 systems, tools, test cases, algorithms, cross-domain reusable assets (Ada run-time kernels and ballistics kernels) , etc. The COE at the LCSEC contains a SEE, a configuration management system, many tools, various test beds and actual target systems, and many algorithms related to its product-lines.

Standard Building Codes - The Army's Technical Architecture, the standard building codes for use in software engineering throughout the Army, is in use at the TACON LCSEC.

Standard Engineering Processes - The LCSEC has legacy processes for several systems and has developed new software for one of its assigned battlefield automated systems using the Cleanroom Software Engineering method. Because of the success of this project, the LCSEC is in the process of adopting Cleanroom as a standard software engineering process. One of the three major components of Cleanroom is the certification method a key element of Software Factory 2001. The LCSEC has begun a direct fire domain analysis, to help define its standard domain architecture.

Process Improvement Program - The LCSEC has been implementing an ongoing process improvement program for several years.

The Obstacles

In this LCSEC example, all of Software Factory 2001's components seem to be in place. What is missing? Only a domain architecture (they have begun to identify and define one) and the cultural shift. Why implementation in 2001? It will take until then to define a domain architecture, institutionalize the product-line approach, and overcome the cultural obstacles.

Domain Architecture

Why is it that the TACOM LCSEC does not already have a domain architecture? Because of a cultural lack of acceptance for the concept. The TADOM LCSEC, being a DoD organization, is also handicapped because nobody owns its product-lines. The DoD does not buy members of a product-line; it buys stand-alone systems. There is no established infrastructure for product-lines. The LCSEC must obtain adoption and support from many acquisition organizations for a single product-line.

Once the LCSEC identifies its standard domain architecture, it must market the architecture to system acquisition managers and convince them to adopt it.

Culture

The list of cultural obstacles that impede the acceptance of anything new is long and well known. The problems range from the met invented herel to the ithis is not the way we have always done itl syndromes. Organizations that attempt to establish product-line architectures are no exception. However, the LCSEC community also has some somewhat unique obstacles. The domain managers (those with the money to make things happen) are currently unable to convince their acquisition managers that a common product-line approach is in their project's best interests. Also, with legacy systems of dissimilar architectures, defining a single or small group of product-line architectures is a huge undertaking. It is hoped that the cultural resistance can be overcome in the next three years.

Conclusion

Software Factory 2001 makes sense because of the many benefits of a product-line orientation:

It can reap the maximum benefits of reuse, including
Reuse of a standard architecture.
Having fewer assets to maintain.
It takes maximum advantage of legacy systems and software.
It creates a backbone to create new members of the product-line.
It is needed to fully capitalize on the contents of the DOE.

Ultimately, Software Factory 2001 encompasses proven concepts that result in increased productivity, reduced risk, increased quality and reliability, reduced development time, and lower costs. In an era of increasing competition and reduced budgets, these advantages cannot be ignored.

About the Authors

S. Wayne Sherer is the chief scientist for the U.S. Army TADOM LCSEC located at Picatinny Arsenal, NJ. He oversees and guides software technology development and transfer into the LCSEC and directs the software acquisition and engineering process improvement efforts. Technology areas include reuse, product-lines, architecture, software support, engineering, quality, process and supporting tools. Mr. Sherer also leads policy, standards, software capability evaluation, contract monitoring and common operating environment efforts. As ARPA STARS deputy program manager (Army), he was focal point for process technology, product-line methods, and acquisition issues. He has spent over 25 years working on Army and Air Force software intensive weapon systems and is an authorized SEI Lead Evaluator.

TADOM LCSEC
U.S. Army TACOM/ARDEC
Attn: ANBTA-AR-FBF-B, Building 352
Picatinny Arsenal, NJ 07006-5000
Voice: 973-724-3531 OSM 680-3531
Fax: 973-724-7850/6020

E-mail: wsherer@pica.army.mil

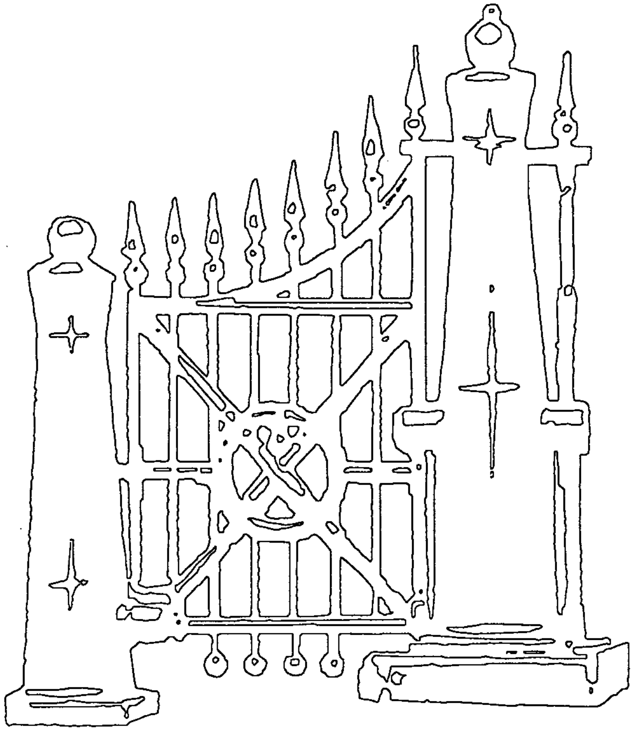
Jack Cooper, president of Anchor Software Management, provides software engineering and management services to the U.S. Army. He is a part-time visiting scientist for software acquisition and risk management at the BEI. He has provided software management support to various U.S. Army battlefield automated systems and to numerous other organizations within DoD. He has been involved in a wide range of software acquisition management-related activities, and has developed several military and DoD standards. He was the chief architect of the Software Acquisition Capability Maturity Model.

Anchor Software Management
422 W. Villa Dunes Drive
Nags Head, NC 27959
Voice: 252-480-8583
Fax: 252-480-2728
E-mail: cooperj@erols.com
75013.2174@compuserve.com

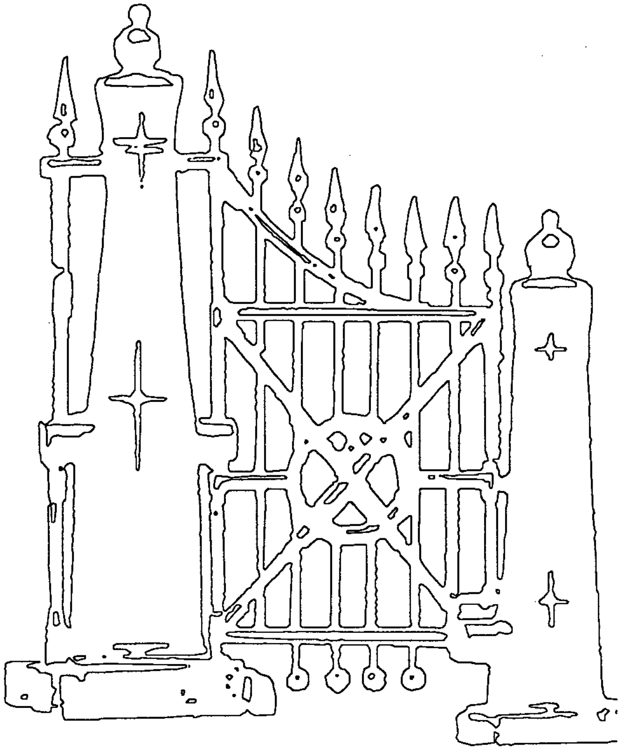
AMC

TACOM

COMMITTED TO EXCELLENCE



Software Factory 2001



S. WAYNE SHERER
TACOM LCSEC
Chief Scientist
DSN 880-3531
wsherer@pica.army.mil

INTRODUCTION

This briefing synthesizes several initiatives:

- **PEO product-lines,**
- **Software reuse,**
- **Architecture,**
- **Reengineering,**
- **Domain analysis,**
- **Joint Technical Architecture,**
- **Greater use of COTS software and**
- **Process improvement;**

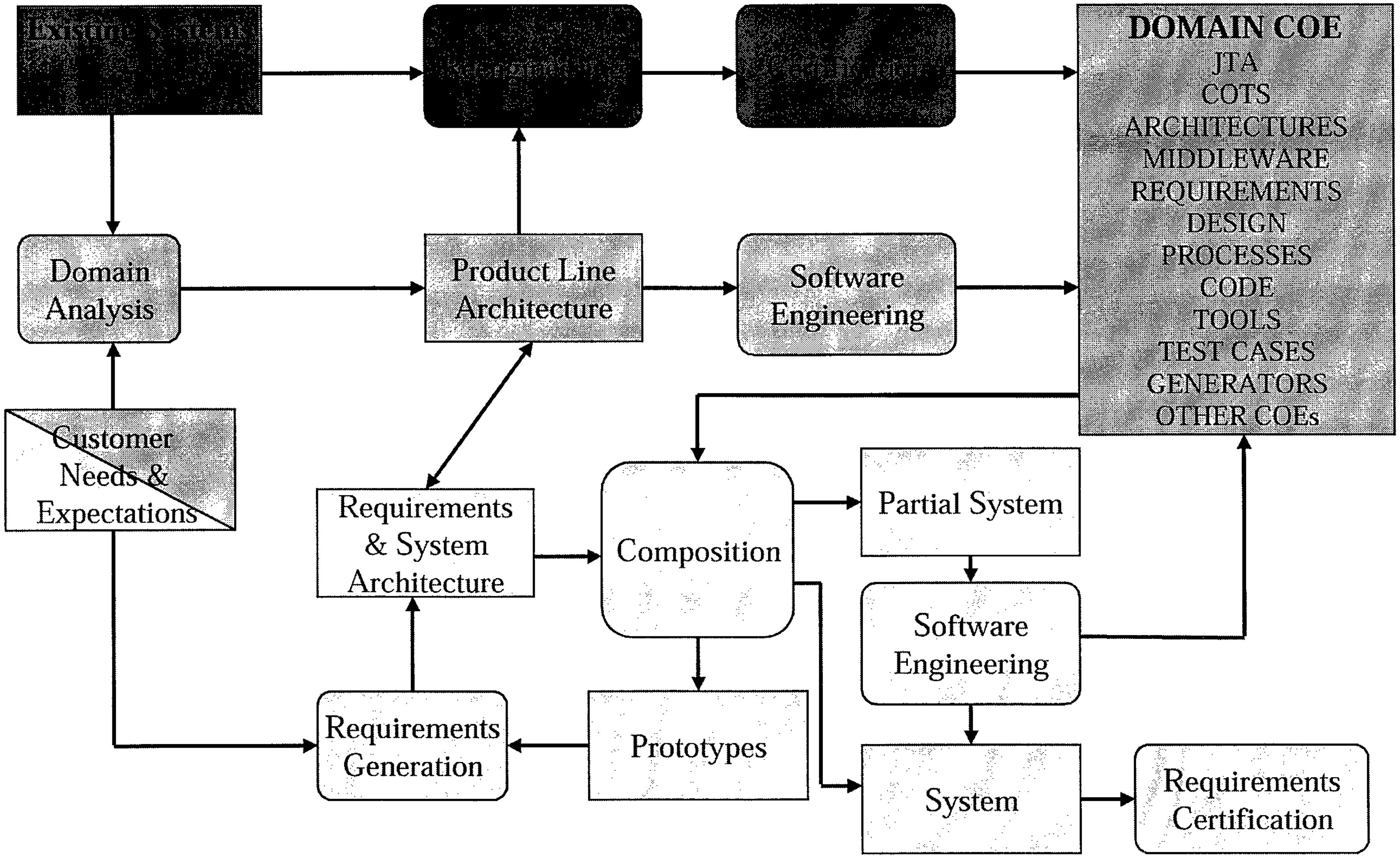
COE & Product Strategy

and presents a process for:

building,
growing,
updating and
maintaining a COE;

and a process for the products assembled
using COE assets.

Software Factory Architecture & Process



Clarifications

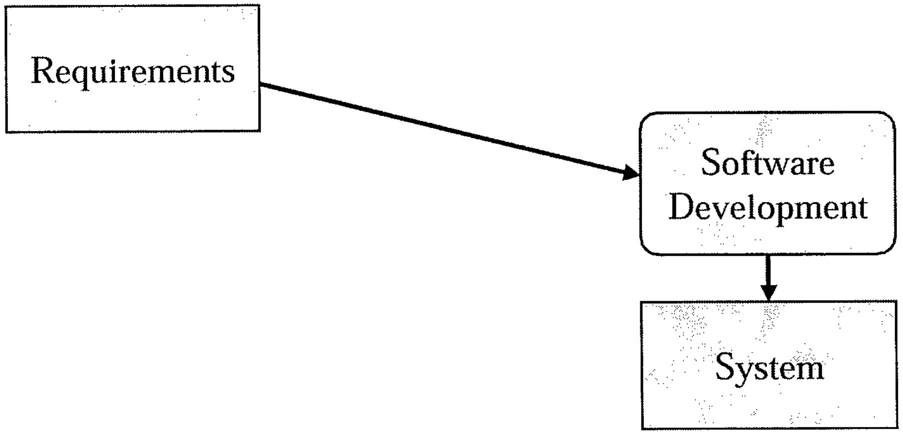
Software Engineering includes: Specification, Development & Certification

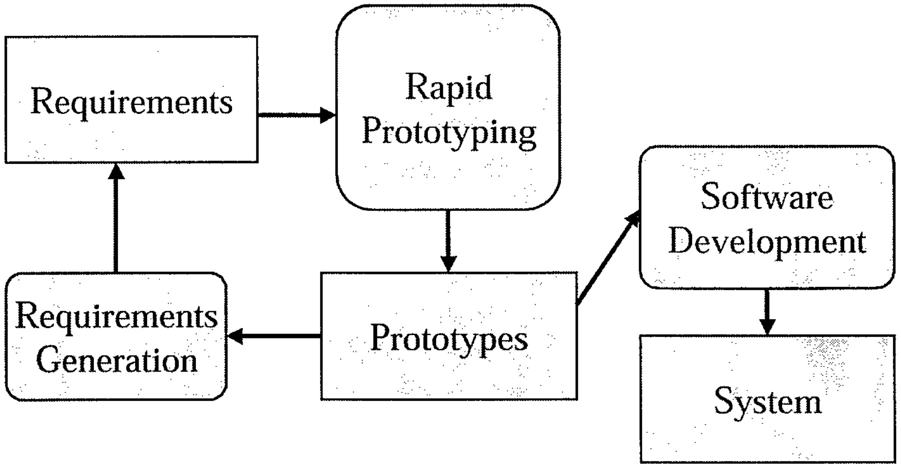
Composition includes: Prototyping, Generation & Linking of components

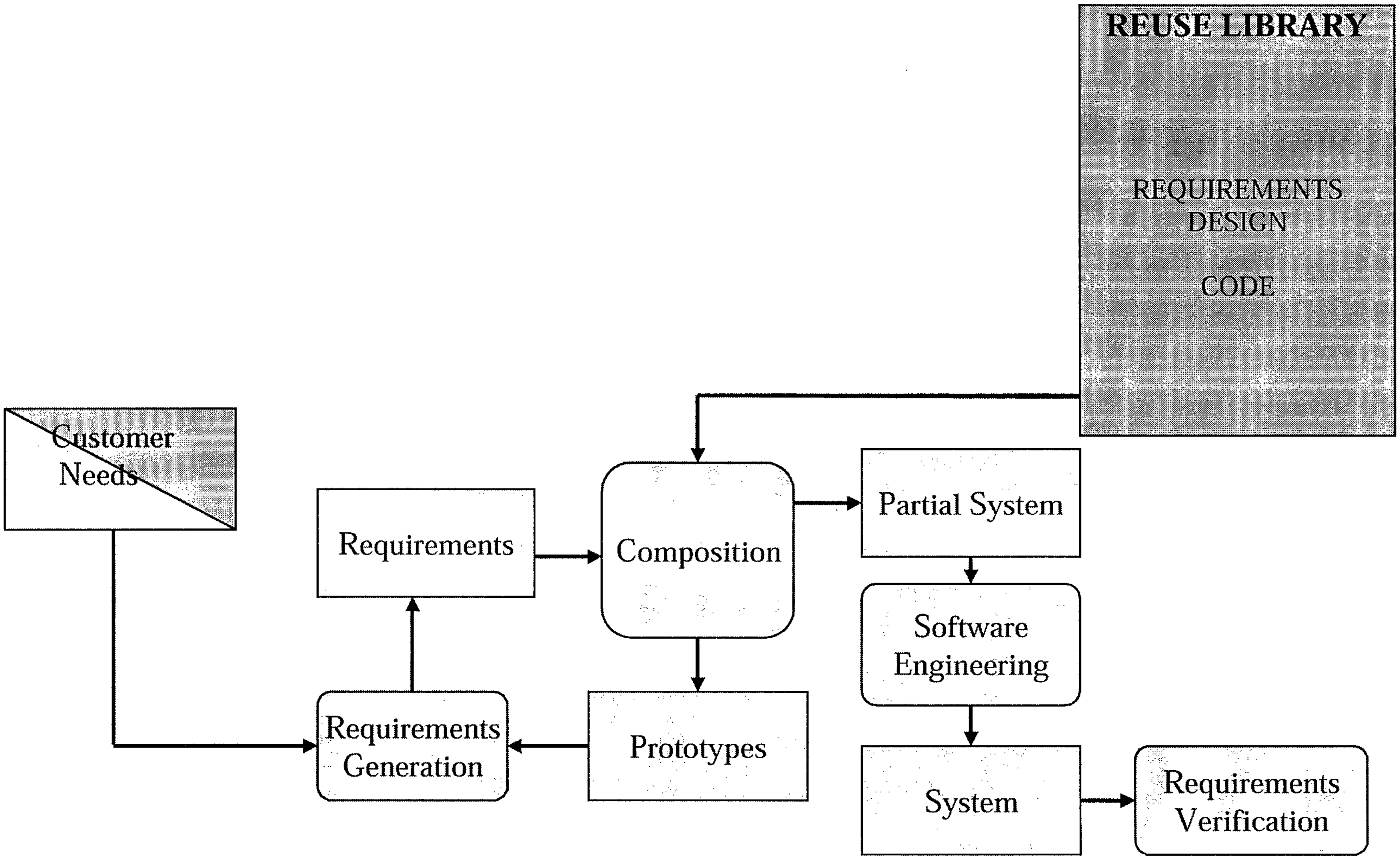
Reengineering includes: Reverse Engineering, Understanding, Restructuring, Reconfiguration & Redevelopment

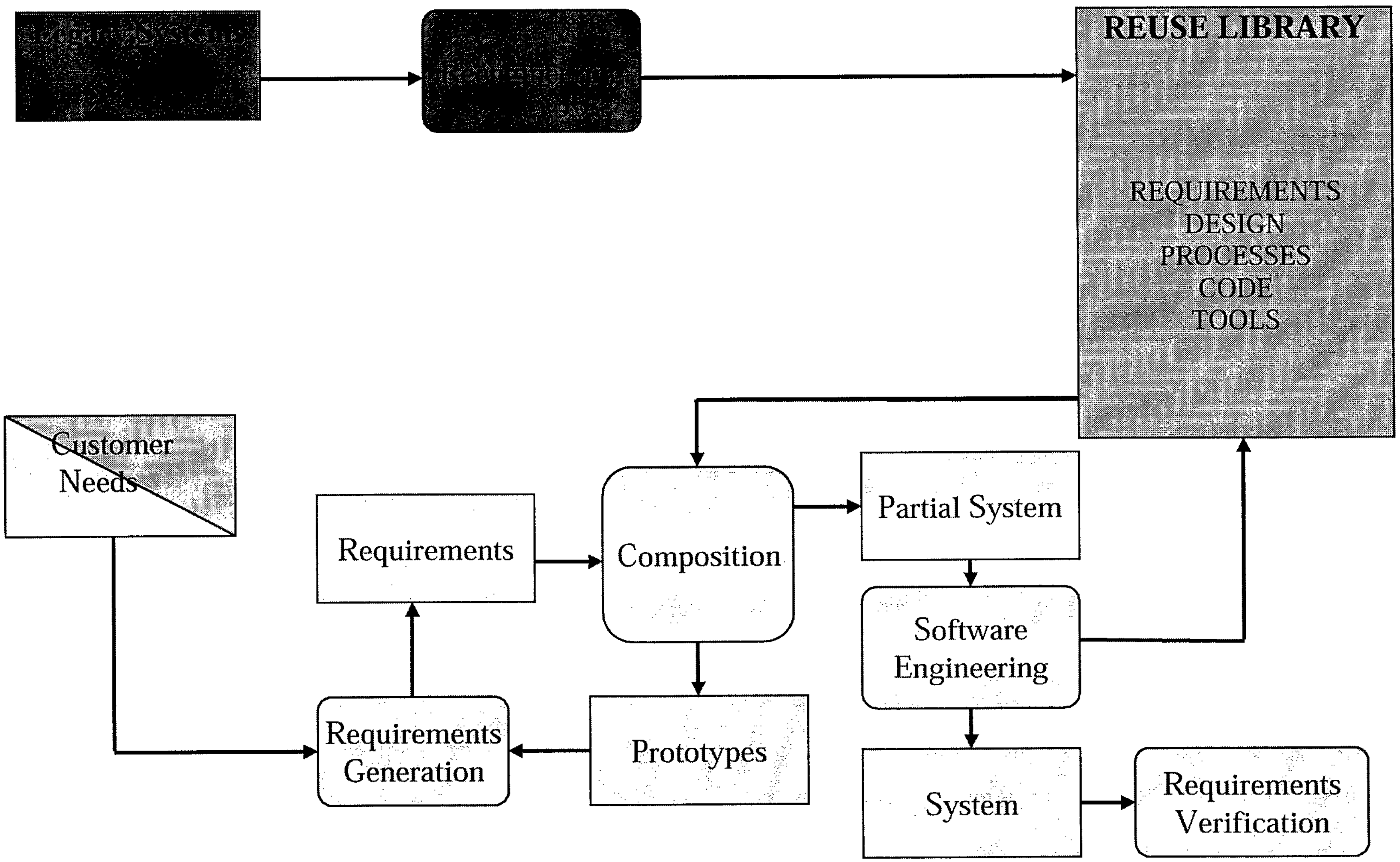
Domain Analysis includes: Understanding, Modeling, Developing & Synthesizing Architectures

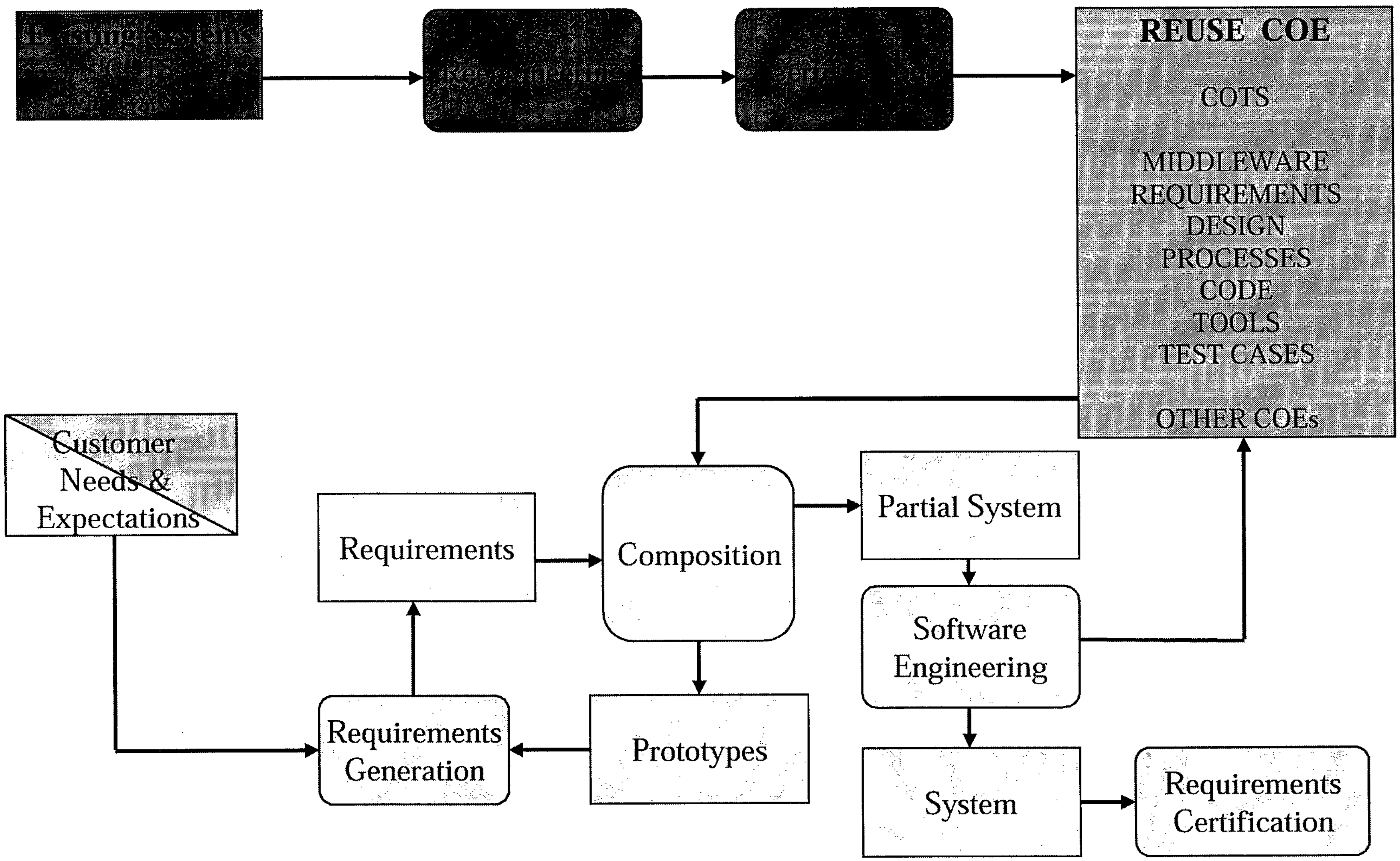
Certification includes: Inspections, Peer Reviews, Audits, Testing and other Evaluations

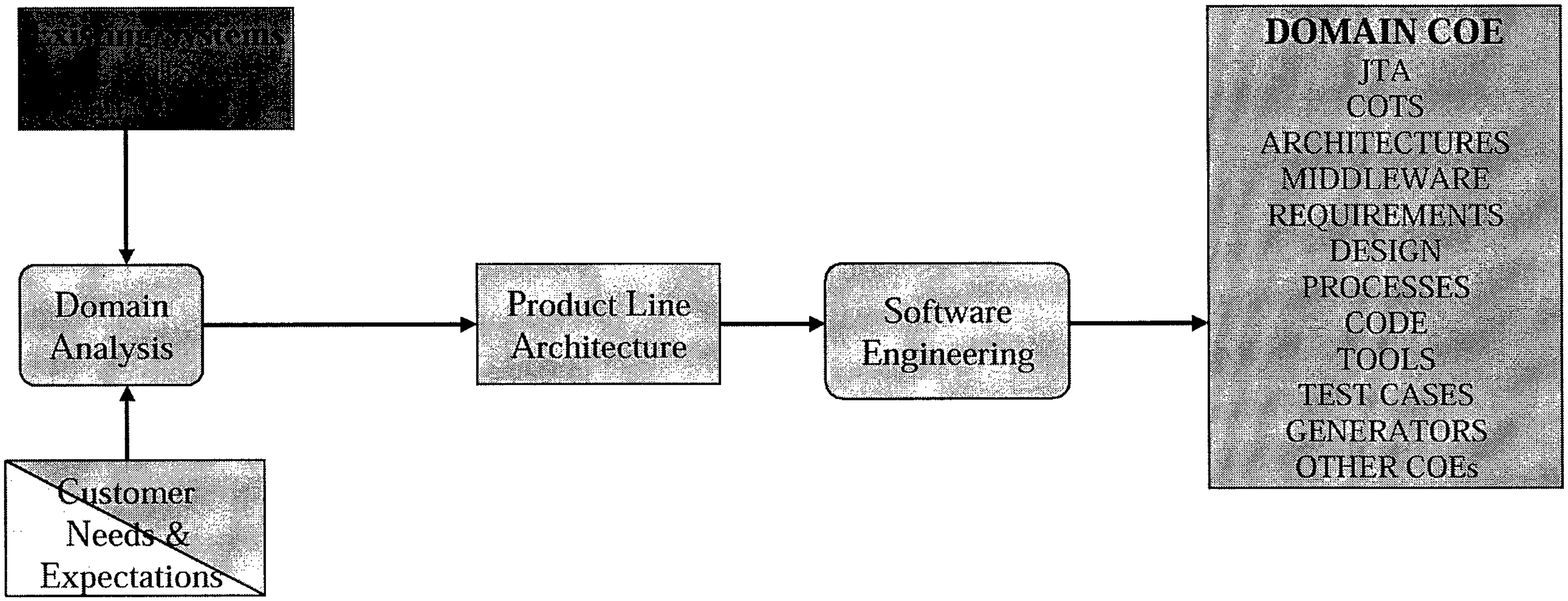

Requirements
Software Development
System


Requirements
Rapid Prototyping
Software Development
Requirements Generation
Prototypes
System

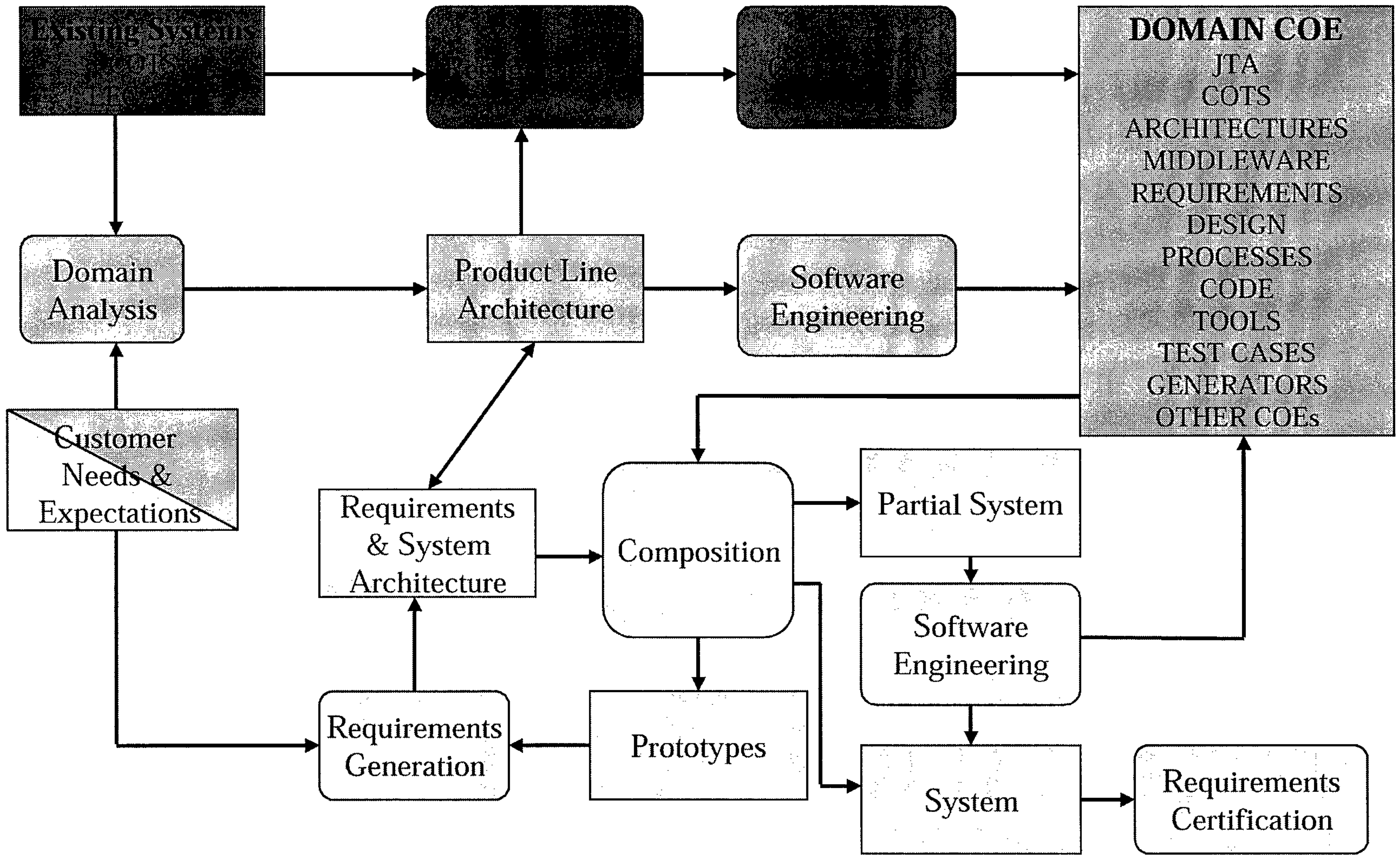

REUSE LIBRARY
REQUIREMENTS
DESIGN
CODE
Customer
Needs
Requirements
Composition
Partial System
Software
Engineering
Requirements
Generation
Prototypes
System
Requirements
Verification


Legacy Systems
Reengineering
REUSE LIBRARY
REQUIREMENTS
DESIGN
PROCESSES
CODE
TOOLS
Customer Needs
Requirements
Composition
Partial System
Software Engineering
Requirements Generation
Prototypes
System
Requirements Verification


Existing Systems
Reengineering
REUSE COE
COTS
MIDDLEWARE
REQUIREMENTS
DESIGN
PROCESSES
CODE
TOOLS
TEST CASES
OTHER COEs
Customer Needs & Expectations
Requirements
Composition
Partial System
Software Engineering
Requirements Generation
Prototypes
System
Requirements Certification


Existing Systems
[illegible]
Domain Analysis
Customer Needs & Expectations
Product Line Architecture
Software Engineering
DOMAIN COE
JTA
COTS
ARCHITECTURES
MIDDLEWARE
REQUIREMENTS
DESIGN
PROCESSES
CODE
TOOLS
TEST CASES
GENERATORS
OTHER COEs

Software Factory Architecture & Process



PRODUCT-LINE OBSTACLES

Stovepipe culture

Ownership and CM

Funding approach

- **Acquisition funding**
- **Maintenance funding**

Domain architecture

Acquisition policy/regulations/laws

BENEFITS

Maximizes reuse

More reliable systems

Lower risks

Increased productivity

Higher quality

Lower costs

An Evaluation of SAR ATR Algorithm Performance Sensitivity to MSTAR Extended Operating Conditions

John C. Mossing[a], Timothy D. Ross[b], and Jeff Bradley[a]

[a]Sverdrup Technology, Inc. Advanced Systems Group 4200 Col. Glenn Highway, Suite 500, Beavercreek, OH 45431
[b]Air Force Research Laboratory, AFRL/SNA, Wright-Patterson AFB, OH 45433

ABSTRACT

Testing a SAR Automatic Target Recognition (ATR) algorithm at or very near its training conditions often yields near perfect results as we commonly see in the literature. This paper describes a series of experiments near and not so near to ATR algorithm training conditions. Experiments are setup to isolate individual Extended Operating Conditions (EOCs) and performance is reported at these points. Additional experiments are setup to isolate specific combinations of EOCs and the SAR ATR algorithm's performance is measured here also. The experiments presented here are a by-product of a DARPA/AFRL Moving and Stationary Target Acquisition and Recognition (MSTAR) program evaluation conducted in November of 1997. Although the tests conducted here are in the domain of EOCs, these tests do not encompass the "real world" (i.e., what you might see on the battlefield) problem. In addition to performance results this paper describes an evaluation methodology including the Extended Operating Condition concept, as well as, data; algorithm; and figures of merit. In summary, this paper highlights the sensitivity that a baseline Mean Squared Error (MSE) ATR algorithm has to various operating conditions both near and varying degrees away from the training conditions.

Keywords: Performance Sensitivity, Automatic Target Recognition, Synthetic Aperture Radar, Extended Operating Conditions, Algorithm Evaluation, Mean Square Error Classifier, MSTAR

1. INTRODUCTION

Synthetic Aperture Radar (SAR) air-to-ground images are collected by various platforms (e.g., the U2, Global Hawk, or F-15E) for various purposes (e.g., reconnaissance or targeting). The collection capacity for such imagery is growing rapidly, and along with that growth is the expanding need for computer-aided or automated exploitation of SAR images. One aspect of the aided/automated exploitation is automatic target recognition (ATR). An ATR algorithm finds target like regions (regions-of-interest (ROIs)) within a SAR image and computes a class (e.g., T72, BTR70, ...) for each ROI. ATR algorithm development typically involves some "training" with example images of known objects.

ATR evaluations are an important basis for technical and programmatic decision making. Technical decisions might include the choice between two alternative approaches to some sub-system or the identification of weak-links within a system. Programmatic decisions might involve competitive down-selects or technology transitions to users. The evaluation of any "trained" system has its challenges, but the evaluation of ATRs seems to be especially problematic because of the constraints on the available data compared to the complexity of the problem.

This paper makes two main contributions. An evaluation methodology is introduced and demonstrated that we believe will lead to results that more reliably support the decision making process. In demonstrating that methodology, the sensitivity of SAR ATRs' to specific (although certainly not all) factors that make the problem complex is reported. The result, we hope, is a better understanding of the SAR ATR problem.

2. BACKGROUND

2.1 MSTAR Program Overview

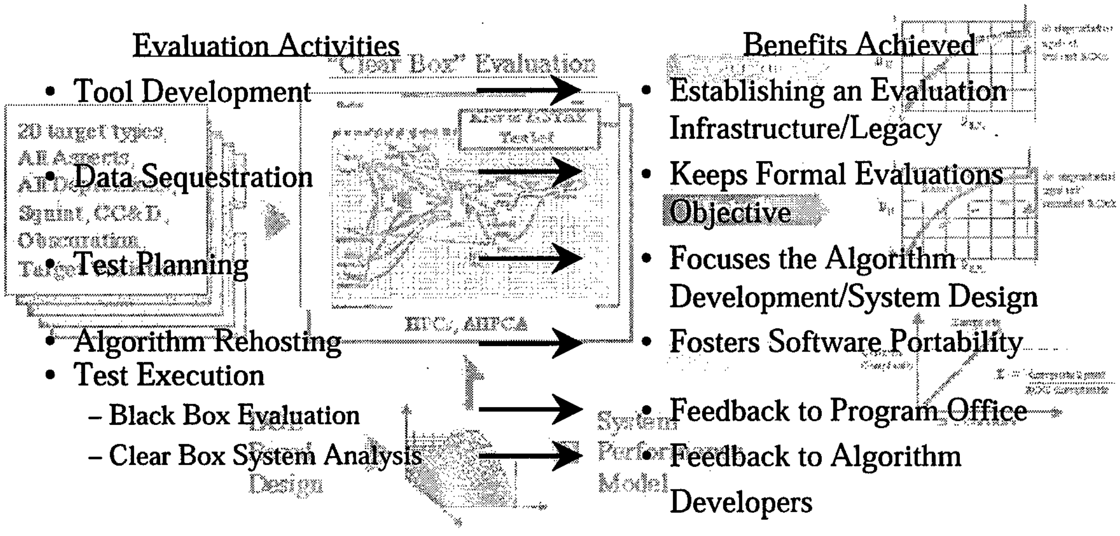
The MSTAR Program Area is focused on the development, integration and evaluation of advanced automatic target recognition (ATR) systems. These systems are capable of high-performance identification of tactical and strategic targets in synthetic aperture radar (SAR) imagery. Such technology is absolutely necessary to achieve dominant battlefield awareness in the face of huge volumes of surveillance imagery. The MSTAR program will provide these automated algorithms and tools for insertion into future advanced technology demonstrations.

The primary goals of the MSTAR Program are to: 1) Develop, integrate and evaluate in the lab a robust and reliable system for the recognition of a twenty target set containing high value tactical and strategic targets: 2) Develop integrated approaches for recognizing targets under variable sensor and deployment conditions: sensor squint, depression and aspect angles; target articulation, configuration, shadow obscuration, terrain layover, and camouflage: 3) Develop and use a modular model-driven ATR architecture to facilitate component evaluation, subsystem re-use and system adaptation to new targets and mission conditions.

The MSTAR program is currently in development. In November of 1997 MSTAR completed the second year of a 3 year base program. The experiments presented in the following sections are a by-product of a November '97 evaluation. An associated team of module developers and a system integrator is developing the MSTAR system under a distributed, collaborative development strategy.[1]

2.2 MSTAR Evaluation

Although this paper focuses on a "baseline" Automatic Target Recognition (ATR) algorithm's sensitivity to extended operating conditions, it is the by-product of an independent evaluation of a more complex ATR system. The baseline algorithm is presented in section 3.2.2. The MSTAR program evaluations are conducted by the Air Force Research Lab (AFRL)/Sverdrup Technology (SvT) Performance Evaluation (PE) team. The MSTAR evaluation is an "in the loop" assessment and feedback process to the Program Office and algorithm developers. This process which is depicted in Figure 1, contributes to the system design, program continuance decision, and technology investment/transition directions. Evaluation activities include "test" planning, data sequestration, evaluation infrastructure/tool development, installing algorithms, "test" execution, analysis, and reporting of results. "Tests" fall into two categories "System Analysis Experiments" and "Evaluations". System Analysis Experiments are conducted with Unsequestered data that the algorithm developers have access to. Evaluations are conducted on Sequestered data that the algorithm developers do not have access to.



Evaluation Activities
- Tool Development
- Data Sequestration
- Test Planning
- Algorithm Rehosting
- Test Execution
  - Black Box Evaluation
  - Clear Box System Analysis

Benefits Achieved
- Establishing an Evaluation Infrastructure/Legacy
- Keeps Formal Evaluations Objective
- Focuses the Algorithm Development/System Design
- Fosters Software Portability
- Feedback to Program Office
- Feedback to Algorithm Developers

Figure 1. MSTAR Evaluation Activities and Benefits.

The Performance Evaluation team reports to the DARPA/AFRL Program Office, The effort is conducted on site at AFRL, WPAFB. Evaluations are conducted periodically throughout the development cycle of the program. These evaluations are goal oriented which focus development efforts.

3. EVALUATION CONCEPTS AND COMPONENTS

3.1 Concept: Extended Operating Conditions

In general, for an ATR algorithm to be successful it must be able to maintain high probability of detection (Pd) and probability of identification (Pid) rates while maintaining a low false alarm rate (FAR) over a variety of *Extending Operating Conditions*. This section attempts to define what the authors mean by Extended Operating Conditions. We start with a scenario of how many ATR algorithms are developed and tested. Collected measured data is commonly needed to develop an ATR. Often a specific data collection is designed to support the development of that particular ATR or data is leveraged off a previous collection with similar goals. Once a data set is available for ATR development, typically it is divided into "train" and "test" sets. Many methods are used to decide which data are used for training and which data are used for testing, some are as simple as placing every other collected sample into the train set with the remaining samples are placed in the test set. In this type of odd/even train/test data partitioning scenario, the ATR algorithm is tested with the test data, which has properties nearly identical to the training conditions. This type of testing is highly discouraged since the results obtained from it would be overly optimistic. The above is an example of what Extended Operating Condition testing is *NOT*. In fact, in the above scenario the testing OCs are as close as one could get to the train conditions without utilizing training data in the testing process. An Extended Operating Condition (EOC) is defined to be at an Operating Condition (OC) "away" from the trained condition. The MSTAR EOCs are presented in section 4.[2,3]. EOC testing is crucial for determining if an ATR algorithm is ready to be fielded.

3.2 Components

Three important ingredients needed to conduct an ATR evaluation are the data, the algorithms, and the performance measures. To conduct an efficient evaluation, obtaining the benefits depicted in Figure 1, it is extremely important that each of these ingredients (data, algorithms, and performance measures) are well understood prior to conducting the evaluation. These components are shown in Figure 2. Sections 3.1 through 3.3 describe them in more detail.

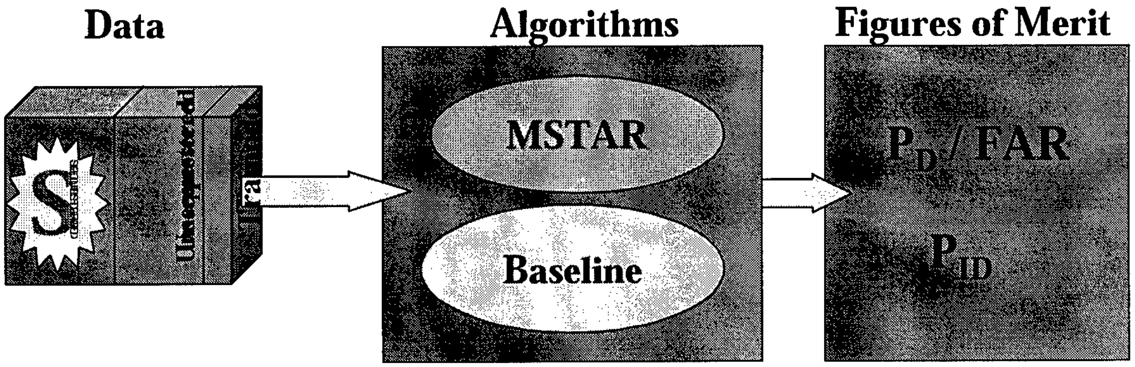


Figure 2. SAR ATR Evaluation Components data, algorithms and performance measures.

3.2.1 The MSTAR Data

To date, the MSTAR program has sponsored three data collections. These collections have resulted in hundreds of thousands of target samples and approximately one hundred square kilometers of clutter data. All of the SAR data was collected at X band, HH polarization, 1'x1' resolution. Figure 3, depicts the "states" of the MSTAR clutter and target scenes. Clutter data was collected at three depressions (15°, 30° and 45°). It was then categorized into three classes, Rural, Sparsely Built-Up, and Built-Up. The target scenes collected were well truthed, contained approximately 25 target types and controlled variations of: target component articulations (guns, turrets, hatches, etc.), revetments, squint, depression, 0-360° target aspects, multiple backgrounds and version variants. The AFRL/SvT Evaluation team has written an "MSTAR Data Handbook for Experiment Planning". The handbook is over 100 pages and is the result of a rather extensive study of all of the MSTAR data. This handbook can be made available by contacting the authors. Once the data was well sorted, sequestration plans were developed, allowing the algorithm developers access to only portions of the data. For more information on the MSTAR data visit the AFRL Data Teams' web site[4].

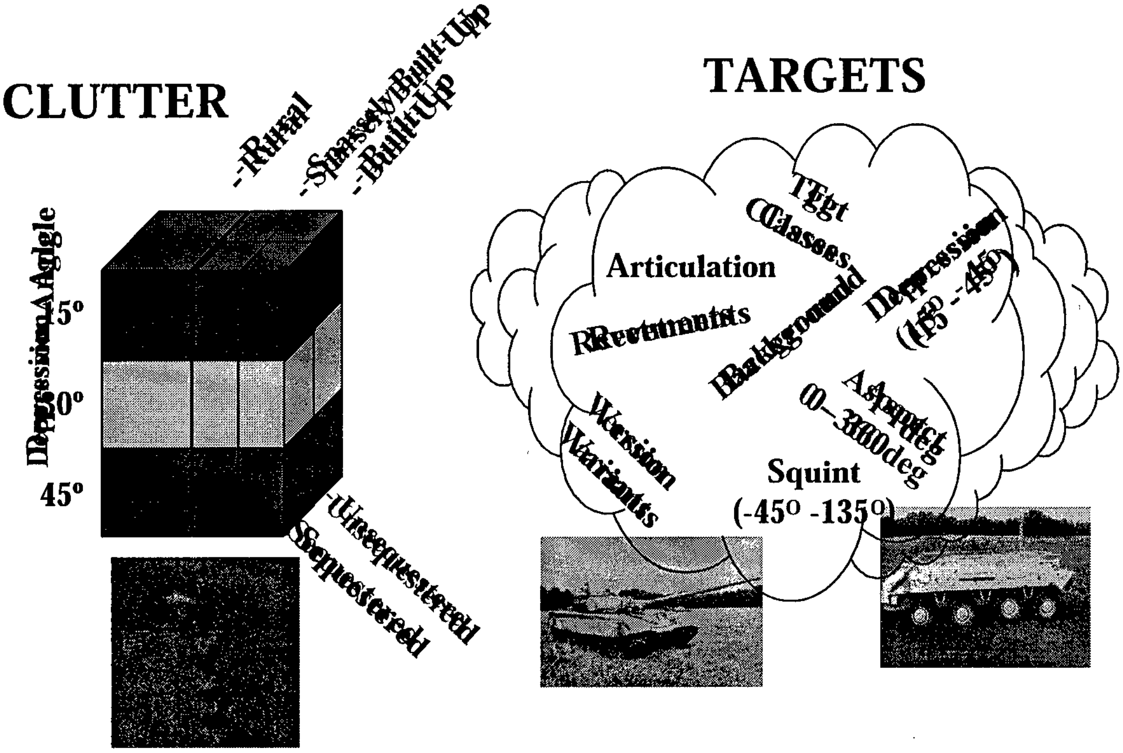


Figure 3. MSTAR data used for testing. All testing discussed in this paper utilized measured MSTAR data. Clutter data was collected at 3 depressions and categorized into three classes. Target scenes contained many EOC states.

3.2.2 Baseline SAR Algorithm Attributes

The baseline SAR algorithm used in this study consisted of three parts: CFAR filter, a target-like/nontarget-like Discriminator, and a Mean Squared Error Classifier. These components and their functions are depicted in Figure 4. The CFAR filter takes as input a full-scene SAR image. The CFAR's job is to locate the Regions of Interests (ROIs) which contain areas of "potential targets". The criteria for something to be a potential target are: a) it be an area that contains some bright pixels and b) and that the cluster of these bright pixels be within two predetermined sized bounding boxes which determine the minimum and maximum size of the targets of interest. The output of the CFAR filter is a row, column location in the full-scene image of the center of the ROI and a CFAR test statistic. The second of the three stages of this ATR Algorithm is the Discriminator. The Discriminator inputs are ROIs as determined by the CFAR. The Discriminator's job is to reject ROIs which aren't "target-like". It does this by training a series of texture, shape, and contrast features. The last of the three stages is the Mean Squared Error template matcher. The MSE's job is to produce a target identification declaration for target ROIs and to reject non-target (clutter) ROIs as non-targets. This stage of the algorithm is also trained. For tests presented in this paper the algorithm was trained across 10 (and 15) target classes. MSE Training for each target class consisted of approximately 225 test chips covering 0°-360° in target aspect at a depression angle of approximately 30°. Training was limited to a single Serial Number (SN) for each target type. A target set was defined which contained many Extended Operating Condition ROIs. These ROIs were sent through the algorithm and an operating point was set such that 90% of the ROIs were detected by the system. Once this operating point was set it was not changed. The algorithm was then tested across many individual and combination of EOC target sets.

Although more complex than the above-mentioned algorithm, the MSTAR ATR algorithm contains the same general detect, discriminate, and classify components. See the AFRL MSTAR Public web page for more detail on the MSTAR architecture[5].

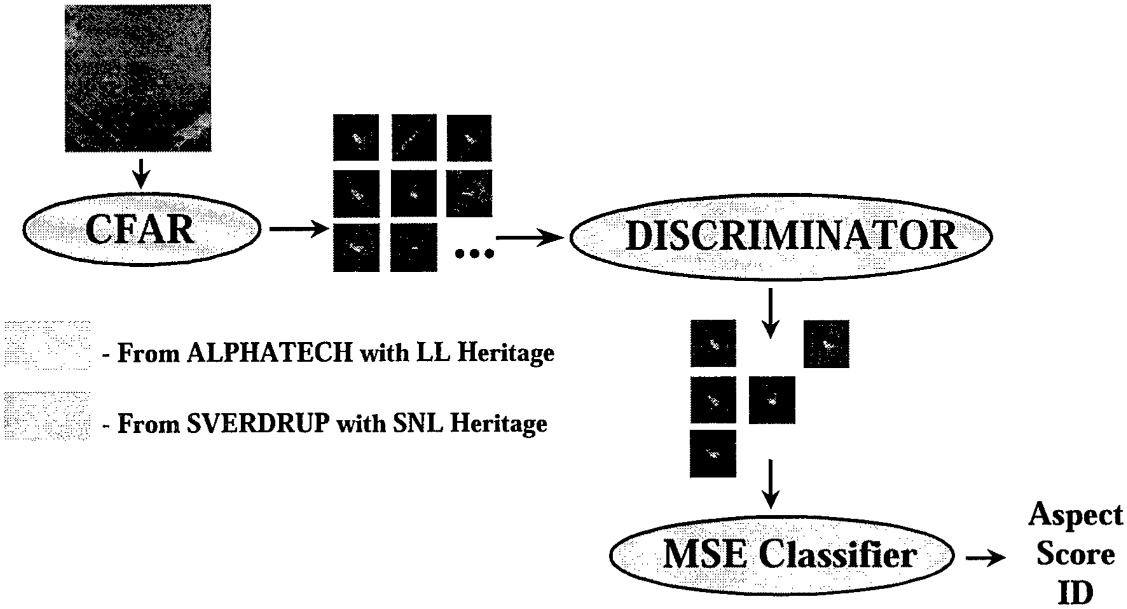

CFAR
...
DISCRIMINATOR
- From ALPHATECH with LL Heritage
- From SVERDRUP with SNL Heritage
MSE Classifier
Aspect
Score
ID

Figure 4. SAR Algorithm Components. The Algorithm consists of three stages a CFAR, Discriminator, and a Classifier.

3.2.2.1 MSE Classifier Metric Definition

In the equations that follow we use the definitions:

$$\dot{x} = \textit{vector form of masked test chip (real-valued)}$$

$$\dot{m} = \textit{vector form of masked mean template (real-valued)}$$

$$\alpha = \textit{scalar scale factor (real-valued)}$$

$$\rho = \frac{\dot{x}^t \dot{m}}{\sqrt{\dot{x}^t \dot{x} \dot{m}^t \dot{m}}} = \textit{correlation between } \dot{x} \textit{ and } \dot{m} \tag{1}$$

The metric implemented by the MSE code is given by

$$MSE = \frac{(\alpha \dot{x} - \dot{m})^t (\alpha \dot{x} - \dot{m})}{\alpha^2 (\dot{x}^t \dot{x})} \tag{2}$$

Where α is given by

$$\alpha = \frac{\dot{x}^t \dot{m}}{\dot{x}^t \dot{x}} \tag{3}$$

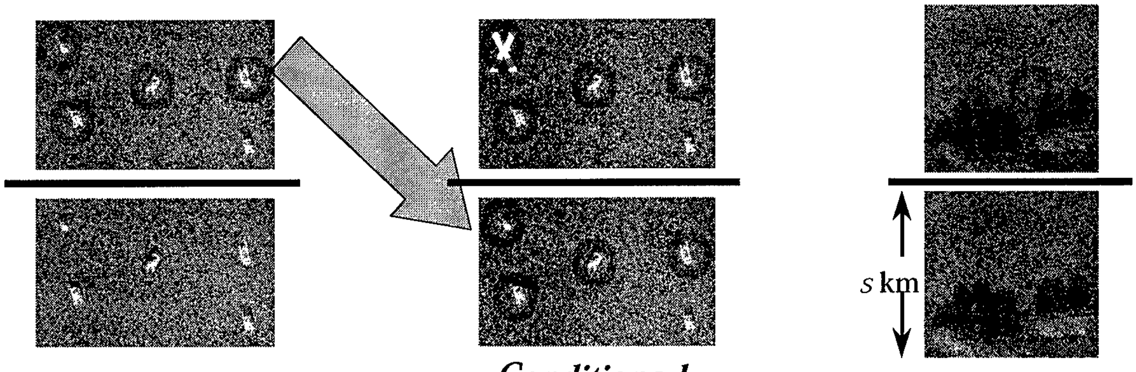
Upon inserting α in the equation for MSE and simplifying

$$\textit{we find} \qquad MSE = \frac{1}{\rho^2} - 1 \tag{4}$$

3.2.3 Performance Metrics Used

The third important ingredient necessary for a successful ATR evaluation involves performance metrics. It is important that the performance metrics be clearly defined and well communicated when presenting results. Figure 5 presents three measures

(Pd, Pid, FAR) and graphically depicts their definitions. When reporting these Figures of Merit (FOMs) an associated confidence interval is presented. We do not go into a detailed description here but do in a "sister" publication[6].



$$P_D = \frac{4}{5}$$

Conditiona l

$$P_{ID} = \frac{3}{4}$$

$$FAR = \frac{\#\mathbf{FA}'s}{s^2\mathbf{km}^2}$$

Figure 5. Performance Measures, Pd and Pid are measured on target scenes, FAR is measured on clutter scenes.

The MSTAR system goal levels for the three performance measures presented above are:

- Pd > 0.9
- Pid > 0.7
- FAR < 0.05 FAs/km^2

Pd and Pid are measured over target scenes while FAR is measured exclusively over clutter scenes. Pid is conditioned on detection (i.e., # of correctly ID'ed targets / # of detected targets).

4. SUMMARY OF ATR TESTING CONDITIONS

4.1 MSTAR II Goals

A variety of evaluation experiments were designed to test algorithm performance against the MSTAR goals. For success the Pd, Pid, and FAR goal levels need to be met across the following EOCs: target versions, depression (15°-45°), number of classes(10 & 15), articulation (M109 and T72 turret), revetment (2' and 5'), configuration, squint (45°-135°), background, aspect (0-360°), serial number (SN).

4.2 Setting an ATR Operating Point

ATR algorithms can be operated at many operating points. A Receiver Operating Characteristic (ROC) curve is the result of plotting a series of ATR operating points. ROC curves are often used to trade off the probability of target detection (Pd) against a False Alarm Rate (FAR) as shown in Figure 6. Once a ROC curve has been generated, a user can use this curve to set an operating point. In this scenario, the operating point defines the Pd and FAR . Once an operating point is set, a confusion matrix can be computed. The confusion matrix is a matrix of fractions. Each column of the confusion matrix represents a possible system decision category (eg. An ATR trained on 10 target types may make 11 decision types. One for each trained target type and an additional decision type for declaring things to be "non-targets"). Each row of a confusion matrix represents a "type" which was presented to the ATR system. Thus the fraction found in cell$_{i,j}$ in the confusion matrix is the number of times the system reported decision_type$_j$ divided by the number of times test_type$_i$ was passed to the system. The diagonal elements of a confusion matrix represent the fraction of the time that the system made correct decisions. The system Pid can be calculated by averaging the diagonal elements. Therefore a confusion matrix with all one's on the diagonal and zeros in all of the off diagonal elements represents a perfect system identification performance of a Pid = 1.0

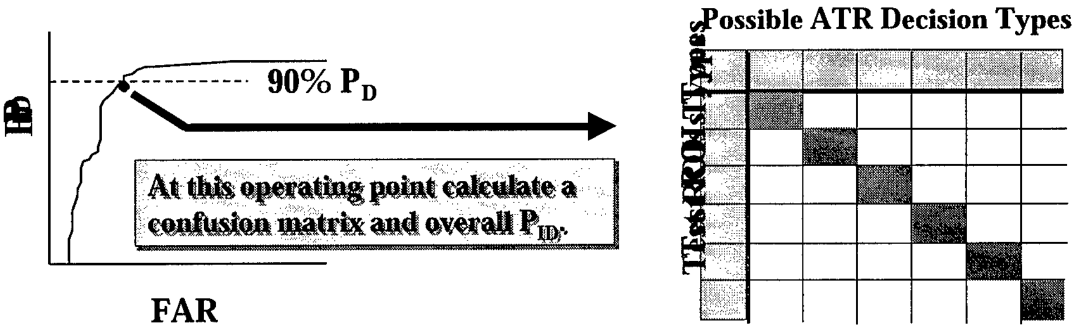

Possible ATR Decision Types
Test ROI Types
PD
90% P_D
At this operating point calculate a confusion matrix and overall P_ID.
FAR

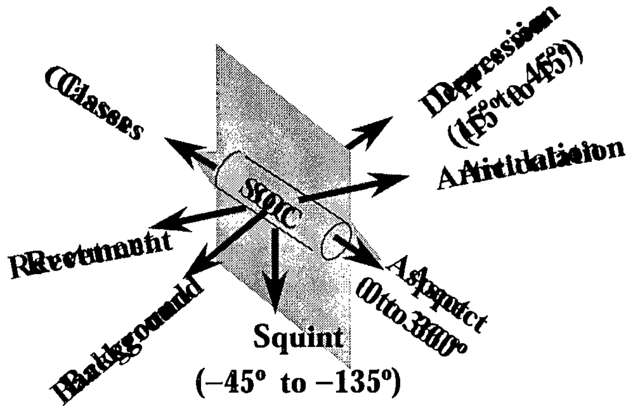
Figure 6. Setting an ATR Operating Point: A ROC Curve and the resulting Confusion Matrix.

For all of the experiments presented in this study the SAR ATR algorithm operating point was set one time **and one time only**. Thirteen experiments will be presented. Each experiment represents algorithm testing over a different target set. Some of these target sets are near the algorithm training conditions and some are not. The ATR algorithm Pd and conditional Pid performance are reported for each experiment. The performance sensitivity as a function of EOCs is then summarized. Note: for the purpose of this study FAR performance sensitivity was not analyzed due to the limited available clutter data.

4.3 Definition of Standard Operating Conditions (nominal conditions)

The Standard Operating Condition (SOC) experiment is defined as the set of operating conditions "very near" training conditions. The SOC test is defined as testing ten target types (M548, M35, Zil131, M2, M113, BTR70, T72, M1, M109, SCUD) across 0°-360° aspect. Seven of the ten target types made use of a single vehicle/serial number during testing. The remaining three target types made use of multiple serial numbers per target type. For the M2s 3 different SNs were tested (MV02GX, MV02FP,T505). For the T72s 3 different SNs were tested (A05, A07, A10). For the M109s 4 SNs were tested (C56, A60, A58, C58). Figure 7 depicts the testing dimensions to be explored. The cylindrical tube represents the standard operating conditions. Each of the labeled vectors off of the tube represent an individual EOC dimension which will be isolated (as best as possible) and tested. The longitudinal axis of the tube spans the "10 target class" and "0°-360° aspect" dimensions. It is represented as a tube as opposed to a line because the SOC test conditions include small deviations in some of the EOC dimensions. The EXP_SOC experiment is defined as test ROIs for the 10 types presented above, sampled across 0°-360° of target aspect, with measured depression angles from 1°-3° away from the ~30° trained depression angle. This EXP_SOC test utilized 630 test ROIs across seventeen different serial numbers. The nominal depression angle for testing was ~30°. For the experiments listed in this paper, unless indicated otherwise, the testing occurred at a depression angle of ~30°.

Figure 7. The Cylindrical tube represents the standard operating conditions. Each vector off of the cylindrical tube represents an EOC dimension. Each of these EOCs will be isolated and tested individually, some in combinations



4.4 Definition of EOC experiments

To measure SAR ATR extensibility across EOCs, EOC experiments are defined. The measured MSTAR data allows for exploration along many of the individual EOC dimensions. The dimensions of interest are depicted in Figure 7. Each vector off of the cylindrical tube represents an EOC dimension. Each of these EOCs will be isolated and tested individually, some in combinations. In this section, we list seven single EOC experiments (i.e., traversing along an EOC dimension) these experiments are listed in Table 1. In addition to these single EOC experiments, five combination EOC experiments (i.e., the test ROIs are "away" from the SOC conditions in more than one EOC dimension at a time) are defined and are listed in table 2. The table is made up of three columns. The first column presents the experiment name. The Second column lists the target types tested and a brief description. The third column enumerates the "test bins" used in each experiment with a description of each bin along with the number of ROIs used in testing.

Table 1. Single EOC Experiment Descriptions

Experiment Name	Experiment Description	Test Bin Descriptions
EXP_ARTIC	M109 (A60,A58,C58,C56) and T72 (a05,132) were tested. There were 13 discrete turret articulation states from-30° to 90° tested.	1) nominal - 320 SOC ROIs, 0° artic 2) low - 410 ROIs, 9°-40° artic 3) high - 560 ROIs > 40° artic
EXP_REVET	T72(A07,A10) and M109s (A60,A58) were tested. These vehicles were in 2' & 4-5' deep revetments	1) nominal, 180 ROIs, no revet 2) half - 60 ROIs, 2' revet, (a10,a58) 3) full - 60 ROIs, 4-5' revet,(a07,a60)
EXP_COLL	T72, M109 and Zil-131s were tested. All training was done with collection 2 vehicles, testing was done across all three MSTAR collections.	1) col2_1- trained in collection 2 tested in collection 1, 60 ROIs 2) col2_2 -trained in collection2 tested in collection2, 270 ROIs 3) col2_3 trained in collection 2 tested in collection 3, 120 ROIs
EXP_DEP	9 target types were tested (the 10 class problem listed in section 4.2 minus the SCUD). The ATR was trained at ~30° depression and tested at a ~15° and ~45° depression angles.	1) nominal, 300 ROIs 2) ~15° depression angle, 270 ROIs 3) ~45° depression angle, 270 ROIs
EXP_15	10 SOC target types tested against a 10 class ATR. Then 5 additional classes were added to the ATR (making it a 15 class classifier). And the 10 SOC target types were tested against a 15 class problem	1) trained with 10 classes tested with 10 classes, 630 ROIs 2) trained with 15 classes tested with 10 classes, 630 ROIs
EXP_SU	M2s and T72s Tested. Testing Sequestered ROIs vs untrained Unsequestered ROIs.	1) Tested on Sequestered, 120 ROIs 2) Tested Unsequestered, 180 ROIs
EXP_SN	M2s & T72s tested. Measuring performance difference between trained SNs but untrained ROIs against untrained SNs and untrained ROIs	1) ROIs from trained SNs but not the training ROIs, 60 ROIs 2) ROIs from untrained SNs, 180 ROIs

Table 2. The Combination EOC experiment Descriptions

Experiment Name	Experiment Description	Test Bin Descriptions
EXP_ART_RET	T72s tested in their nominal state versus T72s in a combination of low or high articulation coupled with being in 2' or 5' revetments.	1) Nominal, 90 ROIs 2) ART+RET, 140 ROIs
EXP_ART_DEP	T72s, M109s tested. At their nominal state versus in a combination of low or high articulation coupled with being 15°s away from the trained depression angle.	1) Nominal, 180 ROIs 2) ATR+DEP, 290 ROIs
EXP_RET_DEP	T72s tested in their nominal state versus T72s in a combination 2' or 5' revetments coupled with being 15°s away from the trained depression angle.	1) Nominal, 90 ROIs 2) RET+DEP, 110 ROIs
EXP_SQU_COL	T72, M109 and Zil-131s were tested. All training was done with collection 2 vehicles. All testing was done using collection 3 imagery.	1) Broadside squint -100°-80°s, 120 ROIs 2) Aft squint,-135°-125°s, 120 ROIs 3) Forward squint,-55°-45°s, 120 ROIs
EXP_SQU_DEP_COL	T72, M109 and Zil-131s were tested. All training was done with collection 2 vehicles. All testing was done using collection 3 imagery coupled with being 15°s away from the trained depression angle	1) Nominal, 120 ROIs 2) SQU+DEP_COL, 240 ROIs

Results are presented in Section 5 for the EXP_SOC experiment; each of the 7 single EOC experiments listed in Table 1; and for each of the five combination EOC experiments described in Table 2.

5. EXPERIMENTAL RESULTS

All of the experiments listed in sections 4.3 and 4.4 were run through the baseline SAR ATR algorithm described in section 3.2.2. A single ATR operating point was set according to the procedure laid out in section 4.2. The operating point was set to achieve a Pd = 0.9 using the union of the test ROIs found in each of the twelve EOC experiments described above. An individual probability of detection (Pd) and probability of correct identification (Pid) were then computed at this operating point.

5.1 ATR Pd Results

Figure 8 depicts the SAR ATR algorithm's Pd performance sensitivity as a function of the thirteen experiments. Eight of the thirteen experiments have near perfect target detection performance. Three of the experiments have detection performance at the goal level of Pd = 0.90. Two experiments, EXP_DEP and EXP_SQU_DEP_COL experienced detection performance degradation below that of the goal level at approximately Pd = 0.80. All in all this SAR ATR algorithm while only trained at conditions near the EXP_SOC test performed quite well against the MSTAR II EOCs in a Pd sense. We don't find this too surprising since all of the EOCs except the revetment EOC were designed to result in almost no information loss from the target signature. The fact that the revetments caused little detection difficulty to this algorithm was pleasantly surprising. Upon review of the SAR signatures (from a ~30° depression angle) of the revetted ROIs, the man made revetments caused little obscuration or layover effects. This was mostly due to the fact that the targets had a few feet of space between the walls of the revetments and the target sides.

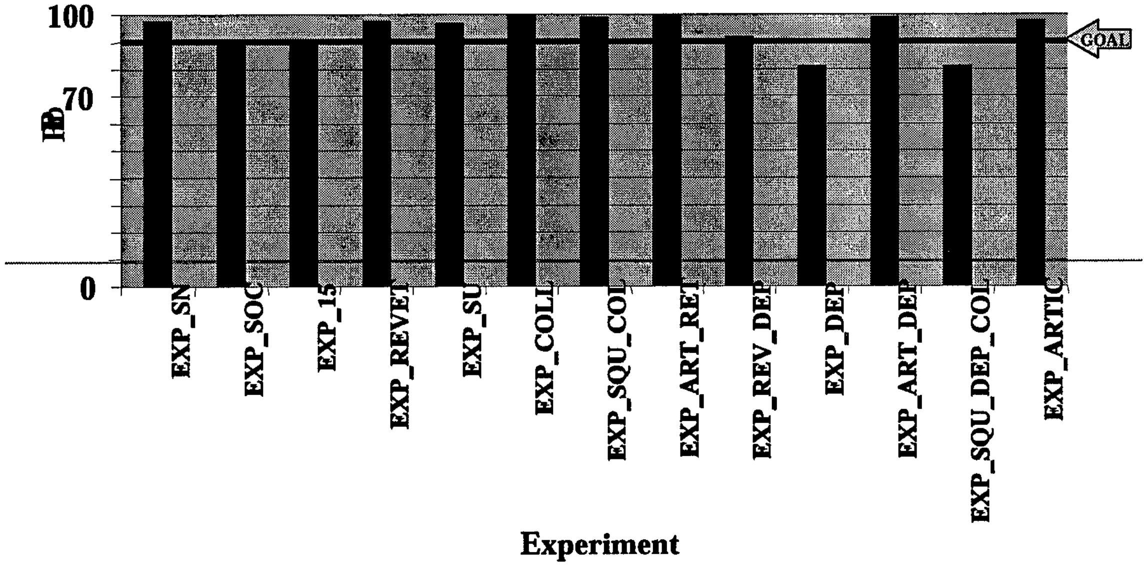


Figure 8. This Bar Chart depicts the Probability of Detection Performance versus Experiment for the baseline SAR ATR algorithm under test. Note: all but two experiments are at or above the goal level of Pd = 0.9.

5.2 ATR Pid Results

Figure 9 depicts the Pid performance for each of the experiments tested. They are sorted into three groups: those at or above the goal level, those below the goal level but greater than 50%, and those below 50%. The confusion matrix in Figure 10 is a result of combining the 12 EOC experiments into a single experiment. The resulting overall Pid = 0.50

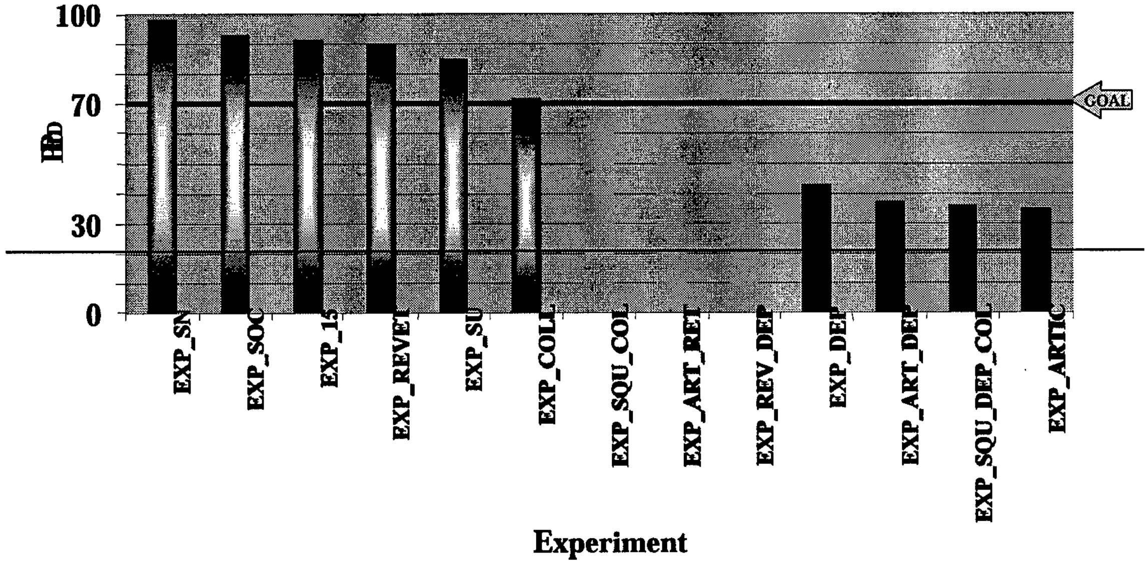
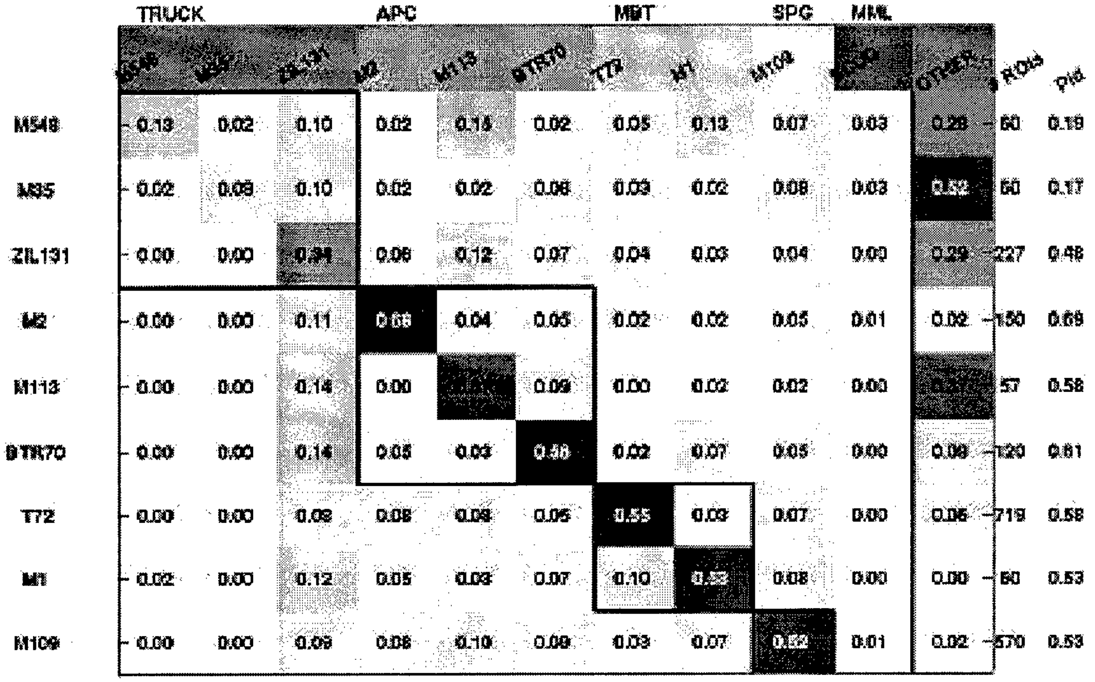


Figure 9. This Bar Chart depicts the Probability of Correct Identification (Pid) Performance versus Experiment for the baseline SAR ATR algorithm under test. Note: Pid performance is very sensitive to EOC conditions. Six of the thirteen experiments were at or above the Pid goal level of 0.70. Seven of the experiments were measurably below the goal level and four of the EOC conditions tested resulted Pid's < 0.50.



	TRUCK			APC			MBT		SPG	MRL			
	M548	M35	ZIL131	M2	M113	BTR70	T72	M1	M109	[illegible]	OTHER	# ROIs	Pid
M548	0.13	0.02	0.10	0.02	0.15	0.02	0.05	0.13	0.07	0.03	0.28	60	0.19
M35	0.02	0.09	0.10	0.02	0.02	0.08	0.03	0.02	0.08	0.03	0.52	60	0.17
ZIL131	0.00	0.00	0.34	0.06	0.12	0.07	0.04	0.03	0.04	0.00	0.29	227	0.48
M2	0.00	0.00	0.11	0.68	0.04	0.06	0.02	0.02	0.05	0.01	0.02	150	0.69
M113	0.00	0.00	0.14	0.00	[illegible]	0.09	0.00	0.02	0.02	0.00	[illegible]	57	0.58
BTR70	0.00	0.00	0.14	0.05	0.03	0.58	0.02	0.07	0.05	0.00	0.08	120	0.61
T72	0.00	0.00	0.09	0.08	0.08	0.06	0.55	0.03	0.07	0.00	0.06	719	0.58
M1	0.02	0.00	0.12	0.05	0.03	0.07	0.10	0.53	0.08	0.00	0.00	60	0.53
M109	0.00	0.00	0.09	0.06	0.10	0.09	0.03	0.07	0.52	0.01	0.02	570	0.53

Figure 10. A confusion matrix representing the Baseline SAR ATR identification performance over the 12 EOC experiments presented in Tables 1 and 2. The overall Pid calculated by averaging the performance across the target types (i.e., the average of the diagonal of the confusion matrix) is Pid = 0.50.

5.2.1 Tests resulting in Pid performance at or above goal level

- EXP_SN The serial number test, EXP_SN, when all EOC dimensions are controlled as much as possible and the Serial Number is the difference from the training and the testing conditions, the baseline ATR's Pid performance was nearly perfect. We cannot conclude from this test that: training on SN X and testing on SN Y (each from the same target class) for all of the X's and Y's in the real world will result in sustained Pid performance as seen in this test. What we can conclude from this test is that if controlling for factors like target version variant differences, target configuration differences, target damage, ... then difference in SN is not the Pid performance driver.
- EXP_SOC When testing an ATR very near the training conditions one would expect excellent Pid performance. It was found to be true with this test.
- EXP_15 When adding five additional potential confuser classes to this ATR, the Pid performance when testing SOC ROIs is not effected
- EXP_REVET The 2' and 5' deep man made revetments used in the MSTAR data collection had very little effect on the SAR signature of the MSTAR targets when viewed from a 30° depression angle and the associated Pid was minimally impacted.
- EXP_SU Testing occurred across ROIs which were not Sequestered but were not trained on by the ATR algorithm and the Pid performance was minimally impacted. This isn't a big surprise this test was designed to measure if in fact the ATR algorithm was over trained on attributes found in the Unsequestered data.
- EXP_COLL When all EOCs dimensions were controlled (with the best of our ability) and the difference was a Collection EOC the ATR algorithm Pid performance dropped to the goal level of Pid = 0.70.

5.2.2 Tests resulting in Pid performance below goal level but above Pid = 0.50

- EXP_SQU_COL The combination of +/- 45° squint from the training condition and test ROIs from a collection different from the training condition resulted in Pid of approximately 0.60 on this MSE based SAR ATR algorithm. Note: all of the training and testing of this algorithm was done in the ground plane. Since the squint angle of the radar is an acquisition parameter it is known and was accounted for when projecting the imagery from the image slant plane into the ground plane.
- EXP_ART_RET Revetments by themselves did not cause Pid performance degradation with the baseline algorithm. However when coupled with the Articulation EOC the Pid ~= 0.60. Note: when articulation was treated as a single EOC, the Pid ~=0.35.
- EXP_RET_DEP Revetments coupled with the depression EOC resulted in a Pid performance of ~=0.60.

5.2.3 Tests resulting in Pid performance below 0.50

- EXP_DEP Depression (i.e., test ROIs +/- 15°s away from the training depression angles) as a single EOC condition resulted in severe Pid performance degradation. The resulting Pid ~= 0.40.
- EXP_ART_DEP Articulation + Depression EOCs when coupled, resulted in a Pid ~= 0.35. That is, approximately 1 out of 3 of the T72s or M109s tested in this experiment were correctly identified by the ATR algorithm.
- EXP_SQU_DEP_COL When the test condition was away from the training condition in three dimensions (Squint, Depression, Collection) the resultant Pid ~= 0.35.
- EXP_ARTIC M109s and T72s were tested at 13 different turret articulation states (-30°-90°). The algorithm was trained at a single articulation state with the turret facing straight ahead (0° articulation). This MSE SAR ATR algorithm had a Pid ~=0.35.

6. SUMMARY

Testing a SAR Automatic Target Recognition (ATR) algorithm at or very near its training conditions yielded near perfect performance in both a Pid and Pd sense. Twelve experiments were set up to measure the algorithm's Pid and Pd sensitivity to Extended Operating Conditions. All of these EOCs (with the exception of revetments) were designed to have little or no impact on the target's SAR signature with respect to "information loss". **As a result the ATR algorithm's ability to achieve high probability of detection (Pd) across these EOC states was not impacted**. As it turned out, the revetment test cases had virtually no impact on the ATR's ability to perform detection or identification. In fact, when the revetment was coupled with either the Articulation or Depression EOC, the Pid of the combination EOC was greater than the Pid for the Articulation or Depression EOC when tested in isolation. This "revetment helping" phenomena shouldn't be given too much weight as the confidence intervals surrounding these Pid's did slightly overlap. **In general, the SAR ATR Pid performance was very sensitive to the EOCs tested.**

7. ACKNOWLEDGEMENTS

This paper is the result of contributions by the entire SvT/AFRL Performance Evaluation team. We would especially like to thank Bob Kotz, Jason Johnson, Ron Dilsavor of Sverdrup Technology and Bill Irving of Alphatech for constructing, hosting and training the SAR ATR algorithm used in this paper. We would like to thank Lannie Hudson and Andy Morrison from Sverdrup Technology, for running the experiments and conducting results analysis. Last but not least, we would like to thank Jason Johnson and Mike O'Connor for assisting in the manuscript preparation.

8. REFERENCES

1. DARPA's MSTAR public web site http://maco.dc.isx.com/iso/battle/mstar.html
2. E.R. Keydel, S.W. Lee, J.T. Moore, "MSTAR extended operating conditions: a tutorial", Proc SPIE, Vol. 2757, pp.228-242, Algorithms for synthetic Aperture Radar Imagery IV, April 1997
3. T. D. Ross, L. Westerkamp, E. Zelnio, T. Burns, "Extensibility and other model-based ATR Evaluation Concepts" Proc. SPIE, Vol. 3070, pp.213-222, Algorithms for synthetic Aperture Radar Imagery IV, April 1997
4. AFRL Data teams' public web site http://www.mbvlab.wpafb.af.mil/public/MBVDATA/
5. AFRL's MSTAR public web site http://134.131.123.60/
6. T.D. Ross, V. Velton, J. Mossing, S. Worrell, M. Bryant, "Standard SAR ATR Evaluation Experiments using the MSTAR Public Release Data Set", Proc. SPIE, vol 3370, SPIE'98 Algorithms for synthetic Aperture Radar Imagery V, April 1998

An Evaluation of SAR ATR Algorithm Performance Sensitivity to MSTAR Extended Operating Conditions

Dr. John C. Mossing, SvT

Dr. Timothy D. Ross, AFRL

Mr. Jeffrey J. Bradley, SvT

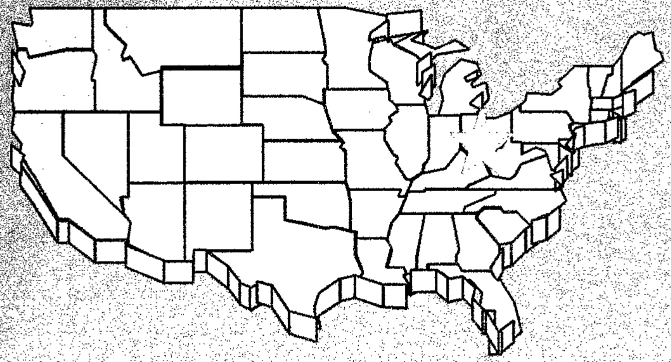
Joint Avionics, Weapons, Systems Support, Software, and Simulation Symposium and Exposition

Sverdrup CORPORATION

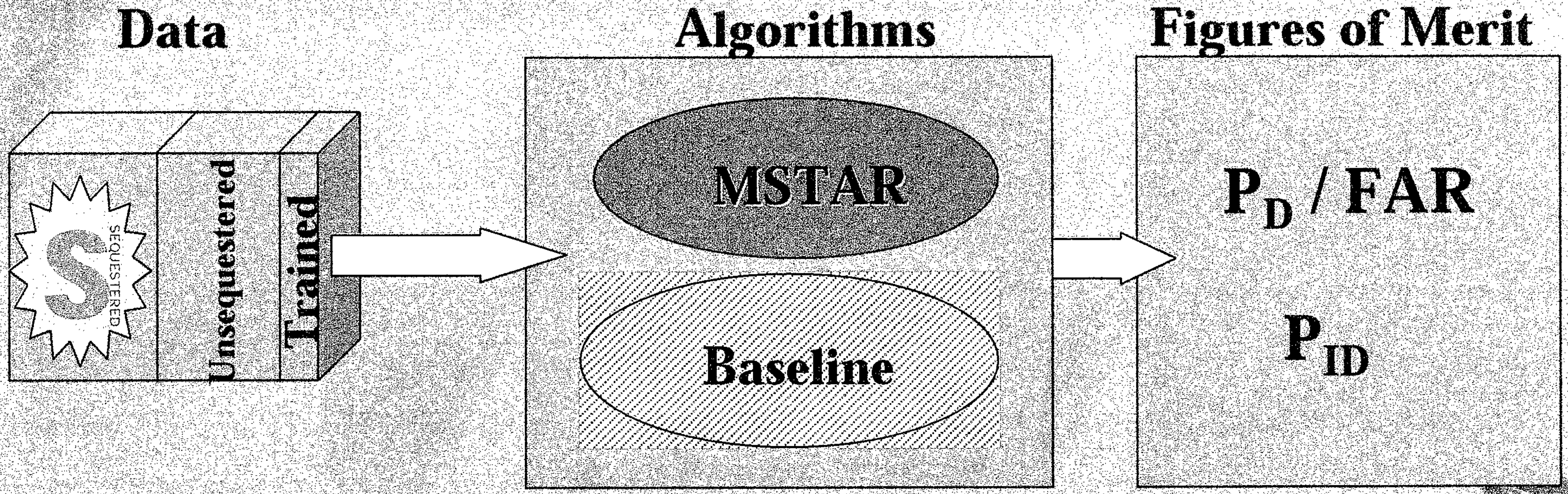
Evaluation Activities

Evaluation Activities		Benefits Achieved
• Tool Development	→	• Establishing an Evaluation Infrastructure/Legacy
• Data Sequestration	→	• Keeps Formal Evaluations Objective
• Test Planning	→	• Focuses the Algorithm Development/System Design
• Algorithm Rehosting	→	• Fosters Software Portability
• Test Execution		
– Black Box Evaluation	→	• Feedback to Program Office
– Clear Box System Analysis	→	• Feedback to Algorithm Developers

Independent Evaluation Strategy



- Goal Oriented Approach
- On-Site at AFRL, WPAFB
- Report to MSTAR PM



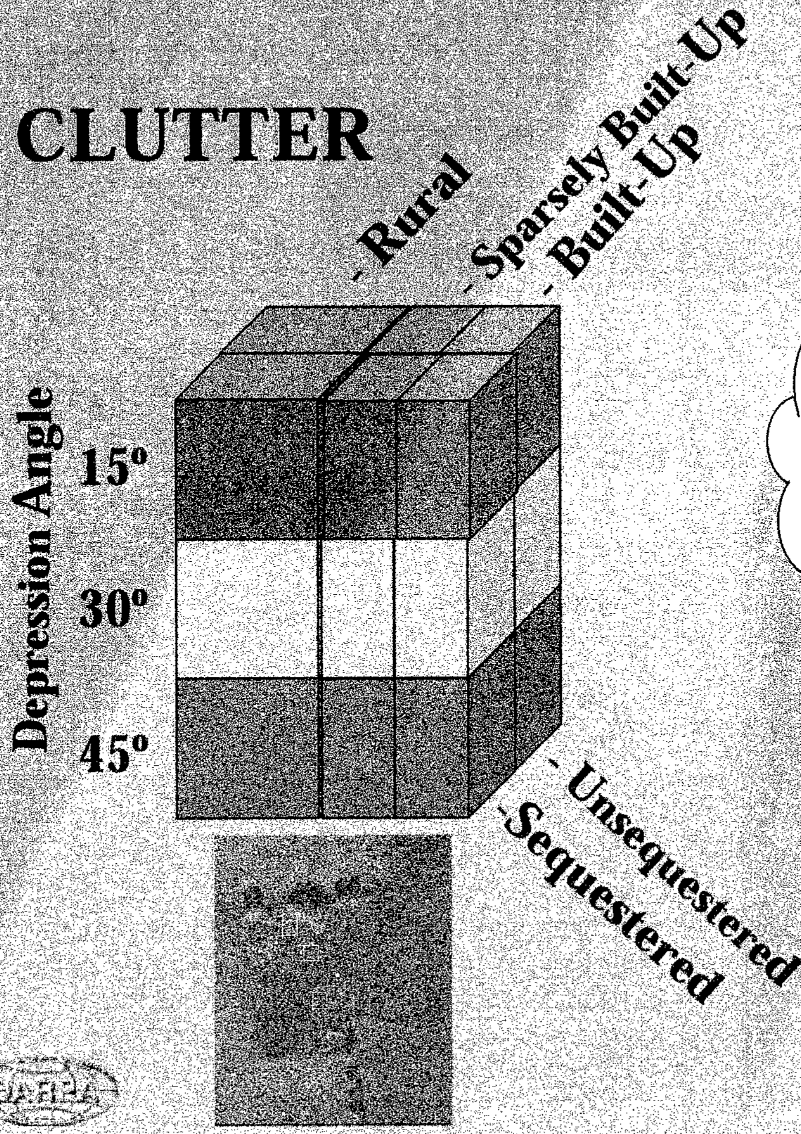
Data

Algorithms

Figures of Merit

Data Sets

CLUTTER



Depression Angle

15°

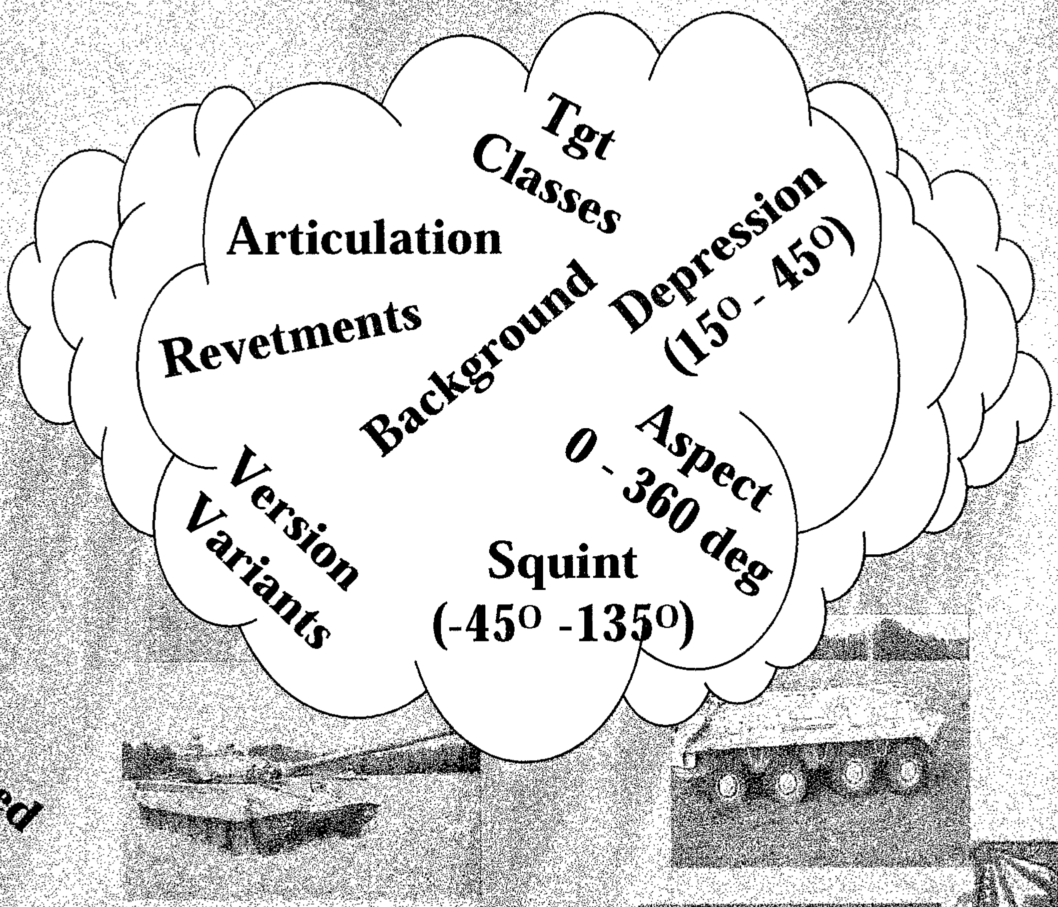
30°

45°

- Rural
- Sparsely Built-Up
- Built-Up

- Unsequestered
- Sequestered

TARGETS



Tgt Classes

Depression (15º - 45º)

Aspect 0 - 360 deg

Squint (-45º -135º)

Version Variants

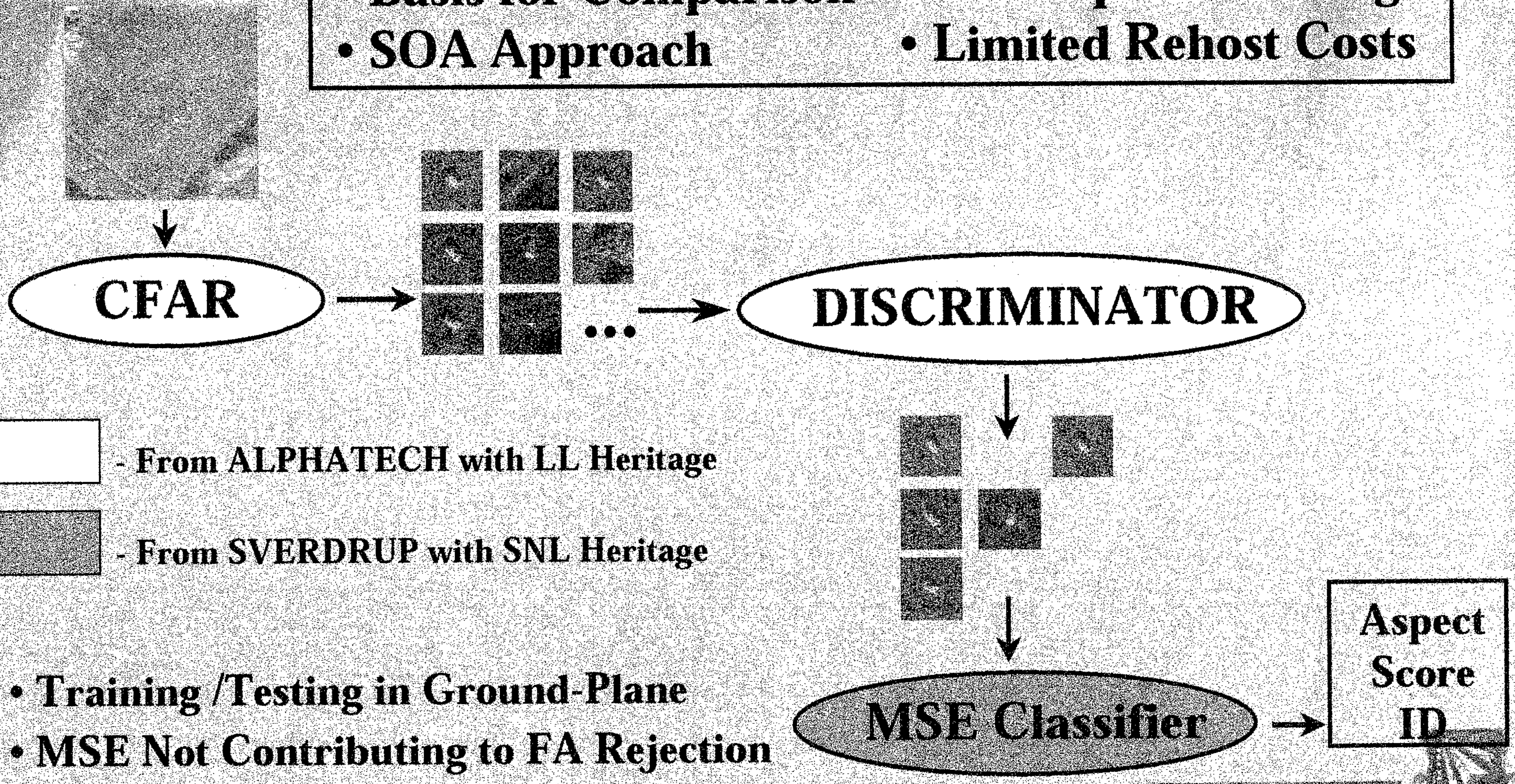
Revetments

Articulation

Background

Baseline ATR System

- Basis for Comparison
- SOA Approach
- Non-expert Training
- Limited Rehost Costs



CFAR → … → DISCRIMINATOR → MSE Classifier → Aspect Score ID

□ - From ALPHATECH with LL Heritage

▨ - From SVERDRUP with SNL Heritage

- Training /Testing in Ground-Plane
- MSE Not Contributing to FA Rejection

MSE metric definition

$\vec{x}$ = *vector form of masked test chip (real – valued)*

$\vec{m}$ = *vector form of masked mean template (real – valued)*

α = *scalar scale factor (real – valued)*

$$\rho = \frac{\vec{x}^t \vec{m}}{\sqrt{\vec{x}^t \vec{x} \vec{m}^t \vec{m}}} = \text{correlation between } \vec{x} \text{ and } \vec{m}$$

The metric implemented by the MSE code is given by

$$MSE = \frac{(\alpha \vec{x} - \vec{m})^t (\alpha \vec{x} - \vec{m})}{\alpha^2 (\vec{x}^t \vec{x})}$$

Where α *is given by*

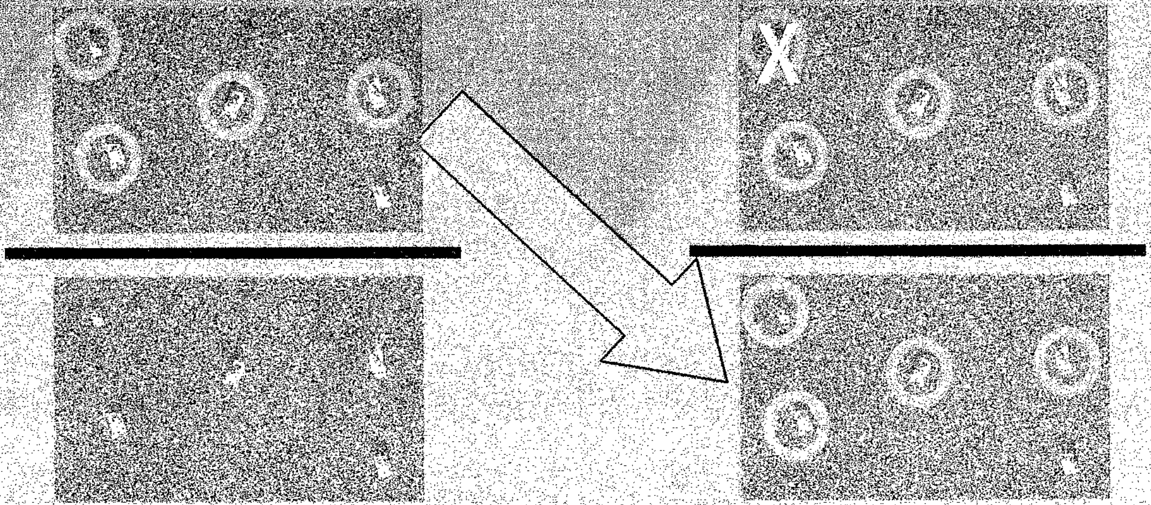
$$\alpha = \frac{\vec{x}^t \vec{m}}{\vec{x}^t \vec{x}}$$

Upon inserting α *in the equation for MSE and simplifying*

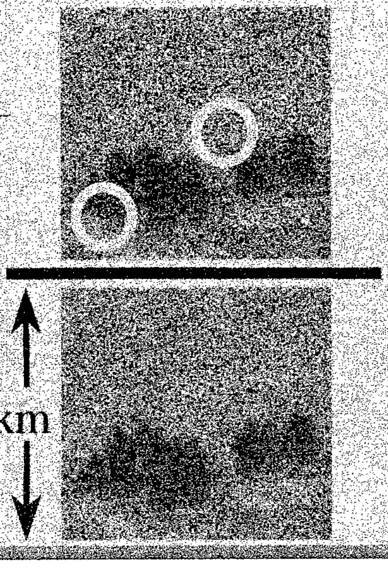
we find $MSE = \dfrac{1}{\rho^2} - 1$

Conditiona l

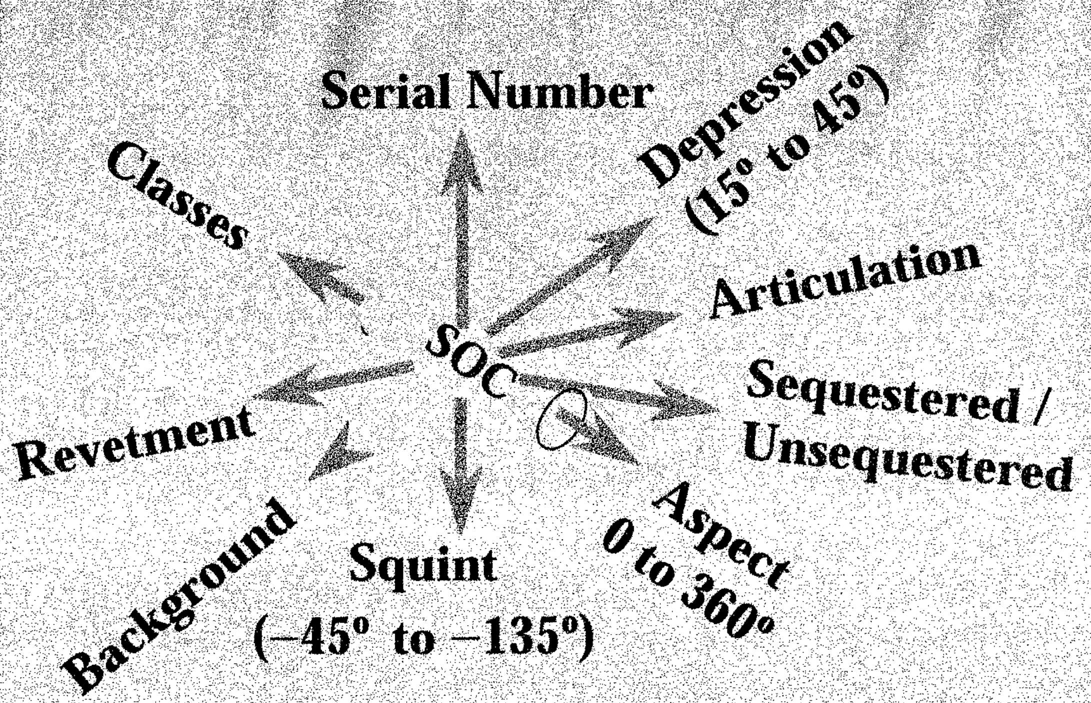
$$P_{ID} = \frac{3}{4}$$



$$FAR = \frac{\#\,\mathbf{FA's}}{s^2\,\mathbf{km}^2}$$



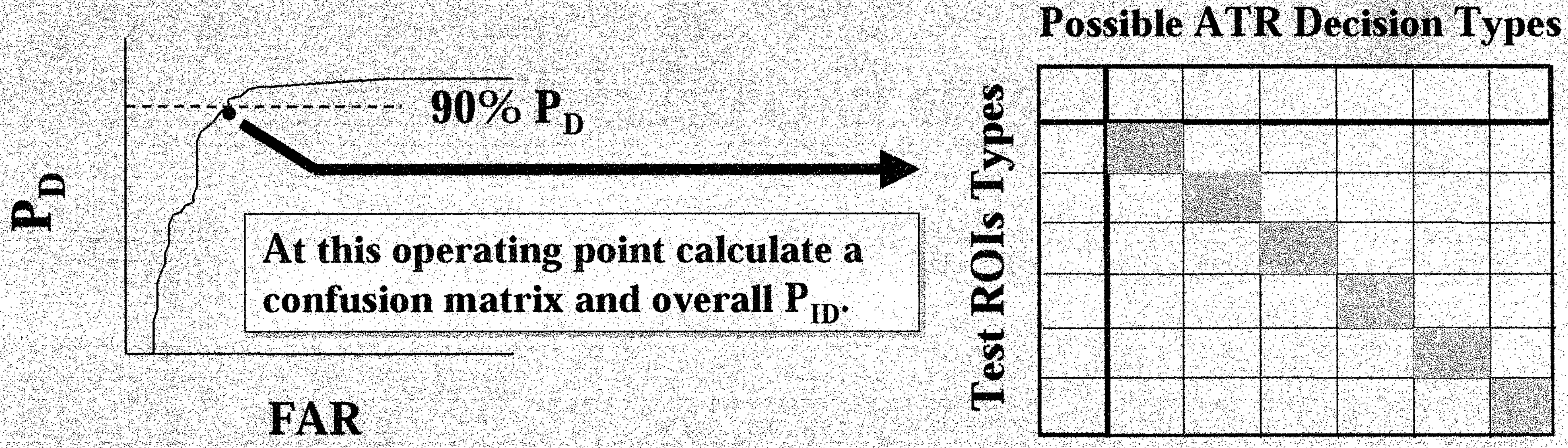
Operating Conditions



- •Combined EOC Conditions
  - •Squint / Background
  - •Articulation / Revetment
  - •Revetment / Depression
  - •Articulation / Depression
  - •Squint / Depression / Background
- •Conditions Excluded from testing
  - •Major version variants
  - •Major configuration variants

Sverdrup CORPORATION

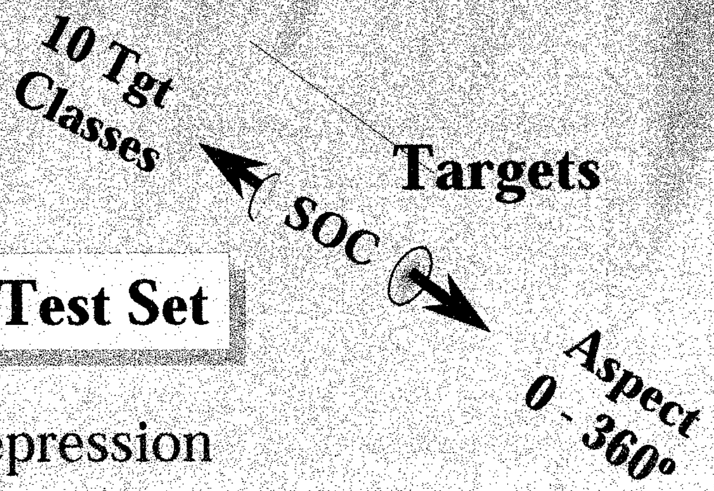
Setting an Operating Point



At this operating point calculate a confusion matrix and overall P_{ID}.

- •Generate a ROC curve against union of all EOC test sets
- •Select operating point at Pd/ROI(S1,EOC) = 0.9
  - •Operating point set exactly 1 time
- •Report Pd and Pid for all experiments at this operating point

SOC Test Results



10 Tgt Classes

Targets

SOC

Aspect 0 - 360°

SOC Test Set

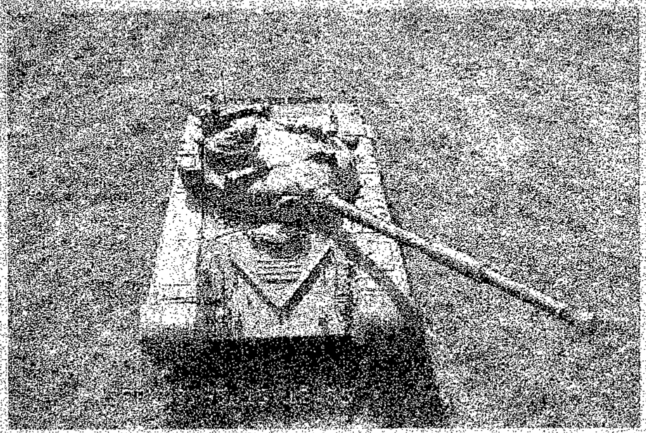
650 ROIs

- System trained ~31 °depression
- System tested ~30 °depression
- 10 class problem
  - Trucks: M548, M35, Zil131
  - APC: M2, M113, BTR70
  - MBT: T72, M1
  - SPG: M109
  - MML: Scud (surrogate)

Pd/ROI(S1,SOC) = 0.9 +/- 0.02

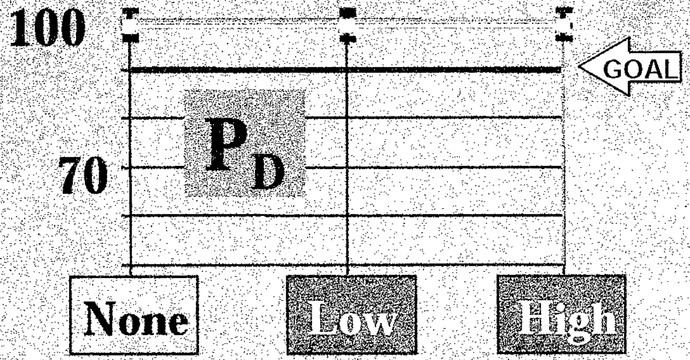
Pid/Ty|D(S1,SOC) = 0.93 +/- 0.02

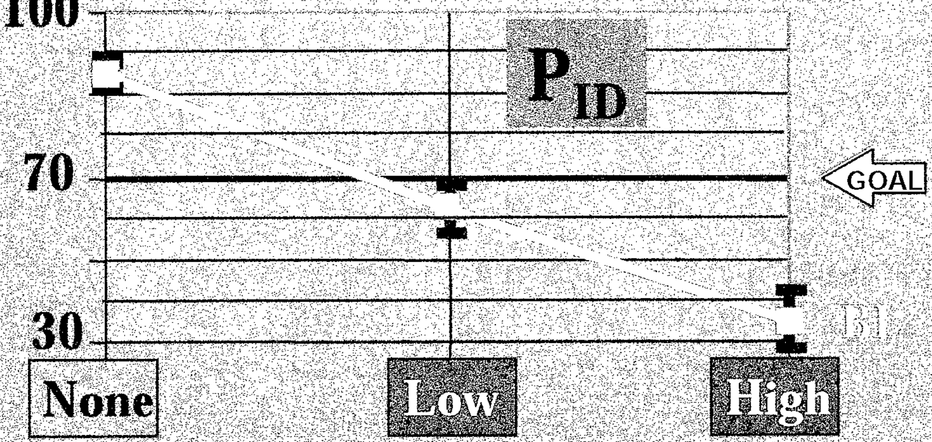
Turret Articulation EOC

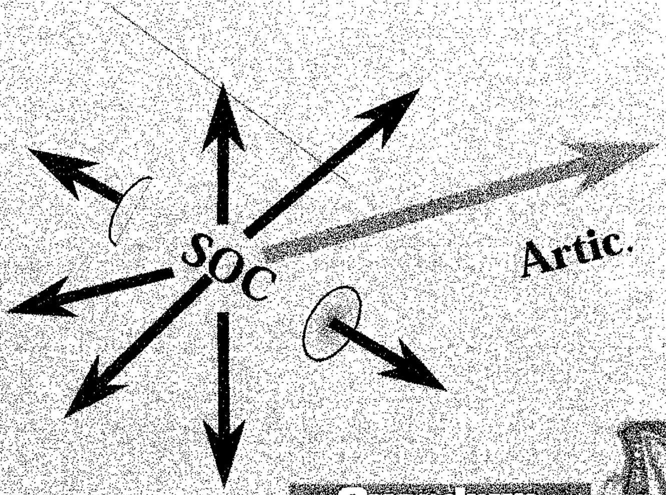


320 ROIs

970 ROIs



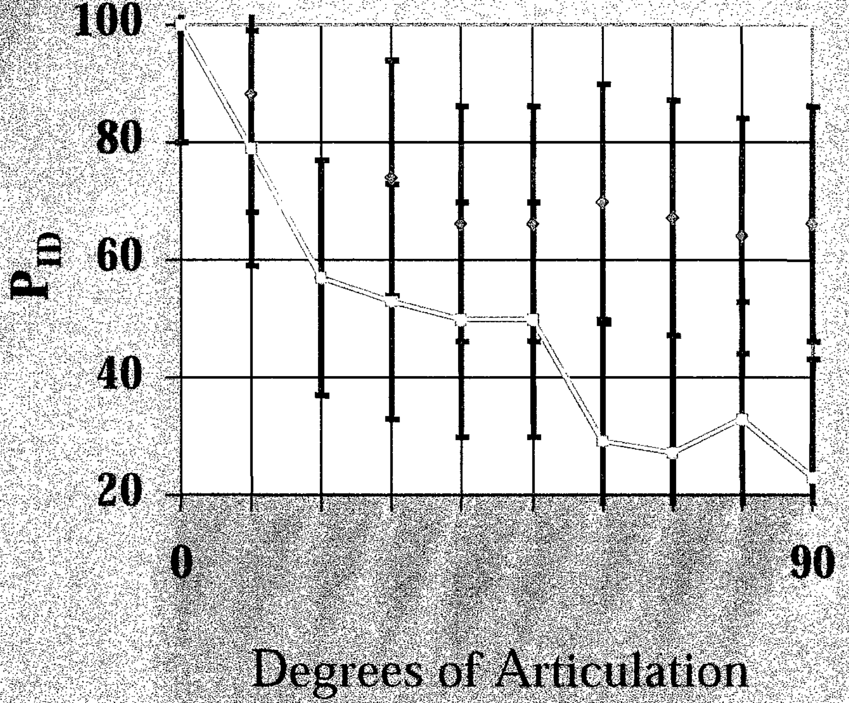




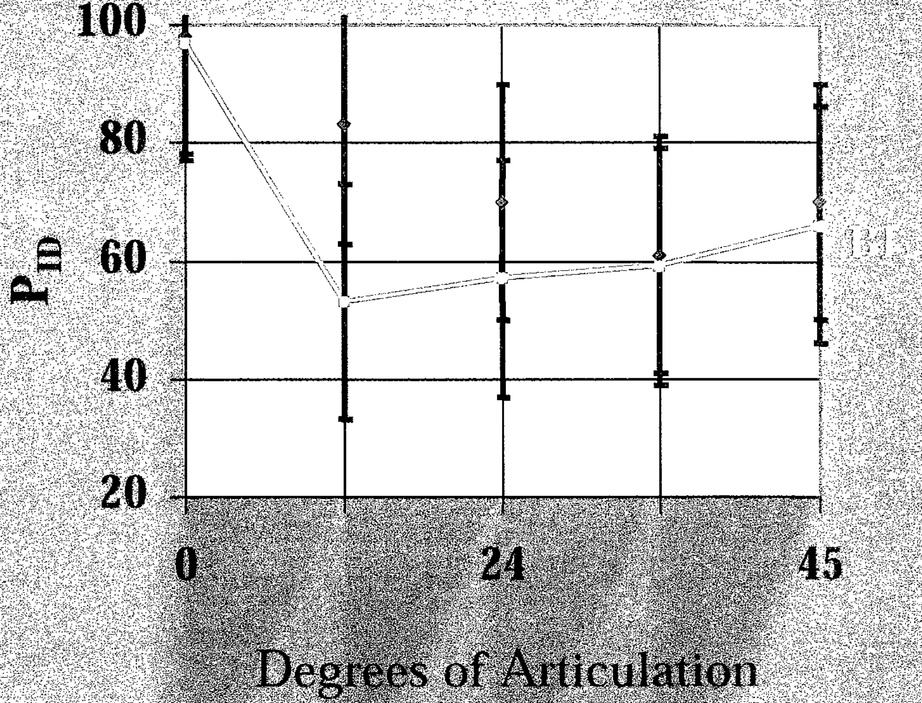
- 10 class system 2 classes tested
    - T72 and M109s
- 13 discrete states -30 to 90 degrees

Turret Articulation EOC

M109 Turret Articulation



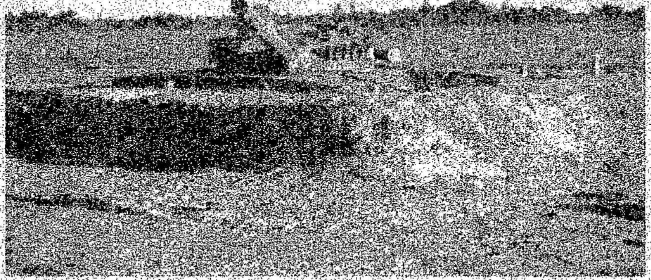
T72 Turret Articulation

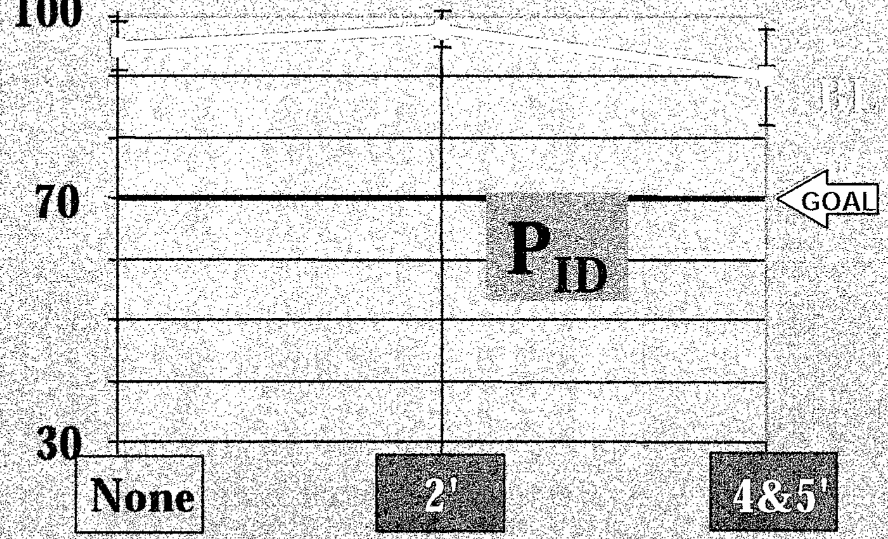


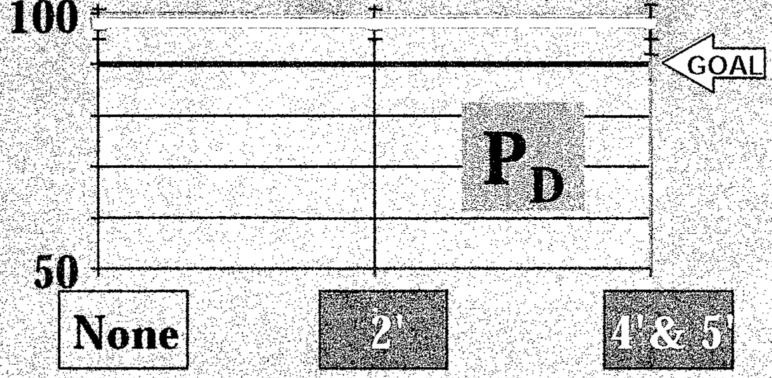
Revetment EOC

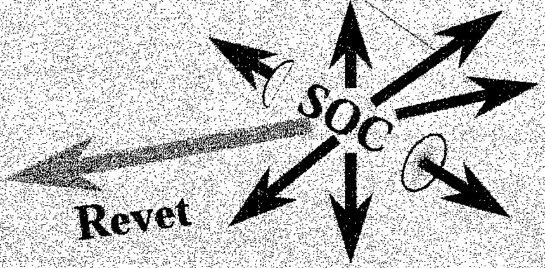
180 ROIs

120 ROIs









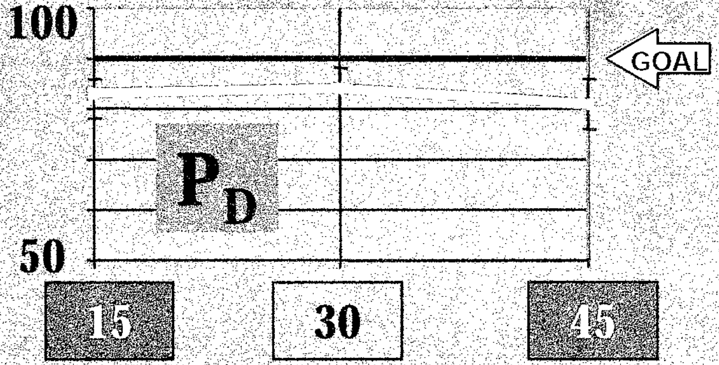
T72 and M109 in 2' and 4&5' revetments

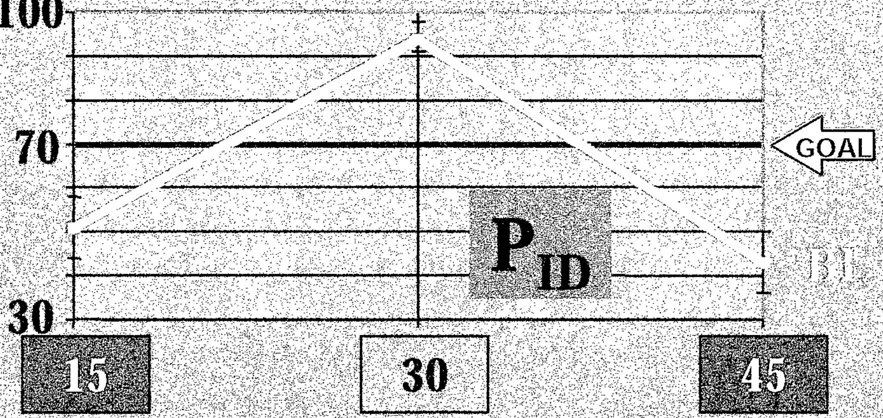
Sverdrup CORPORATION

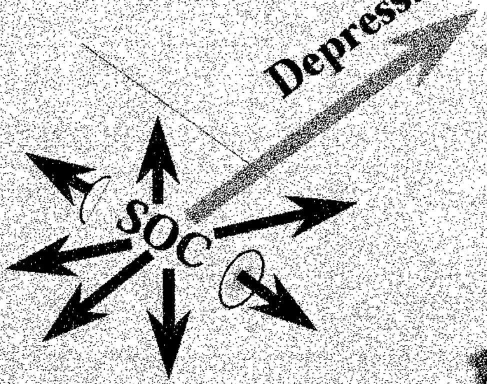
Depression EOC

300 ROIs

640 ROIs





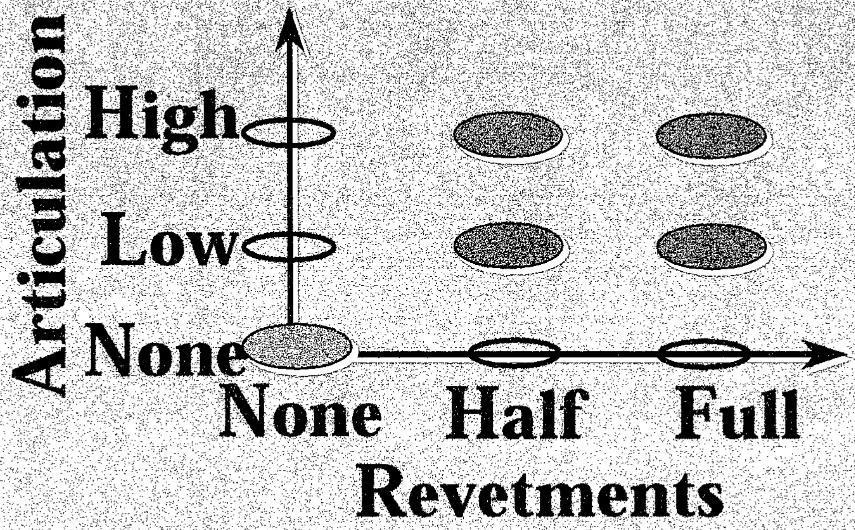


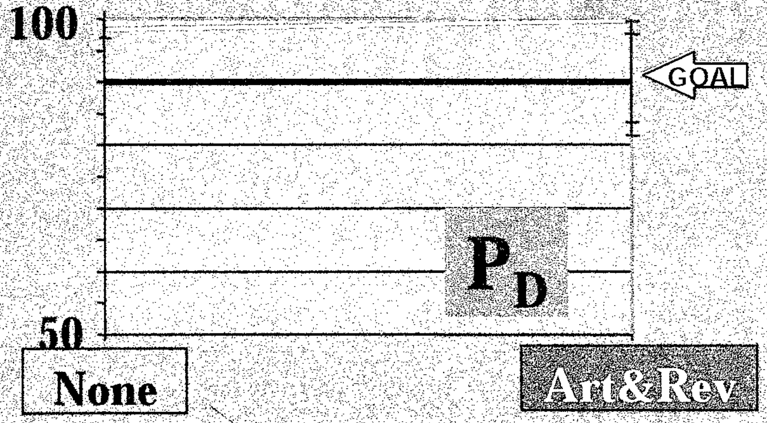
- 10 class system 9 classes tested
- Trained at 30 degrees tested +/- 15

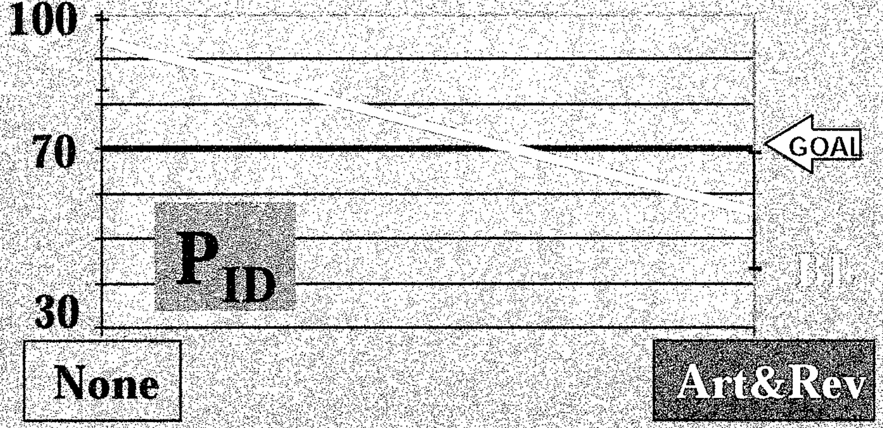
Articulation/Revetments EOCs

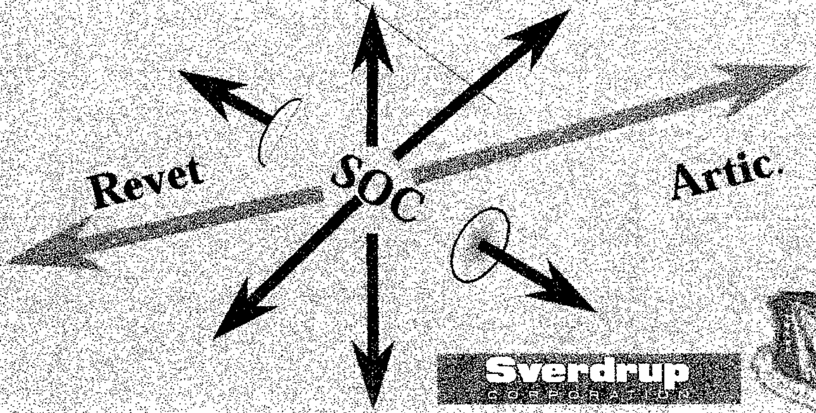
60 ROIs

140 ROIs









T72s Tested

Sverdrup CORPORATION

Description of Experiments

Experiment Name	Experiment Description	Test Bin Descriptions
EXP_ARTIC	M109 (A60,A58,C58,C56) and T72 (a05,132) were tested. There were 13 discrete turret articulation states from-30Υto 90Υtested.	1) nominal - 320 SOC ROIs, 0Υartic 2) low - 410 ROIs, 9Υ-40Υartic 3) high - 560 ROIs > 40Υartic
EXP_REVET	T72(A07,A10) and M109s (A60,A58) were tested. These vehicles were in 2' & 4-5' deep revetments	1) nominal, 180 ROIs, no revet 2) half - 60 ROIs, 2' revet, (a10,a58) 3) full - 60 ROIs, 4-5' revet,(a07,a60)
EXP_COLL	T72, M109 and Zil-131s were tested. All training was done with collection 2 vehicles, testing was done across all three MSTAR collections.	1) col2_1- trained in collection 2 tested in collection 1, 60 ROIs 2) col2_2 -trained in collection2 tested in collection2, 270 ROIs 3) col2_3 trained in collection 2 tested in collection 3, 120 ROIs
EXP_DEP	9 target types were tested (the 10 class problem listed in section 4.2 minus the SCUD). The ATR was trained at ~30Υ depression and tested at a ~15Υand ~45Υdepression angles.	1) nominal, 300 ROIs 2) ~15Υdepression angle, 270 ROIs 3) ~45Υdepression angle, 270 ROIs
EXP_15	10 SOC target types tested against a 10 class ATR. Then 5 additional classes were added to the ATR (making it a 15 class classifier). And the 10 SOC target types were tested against a 15 class problem	1) trained with 10 classes tested with 10 classes, 630 ROIs 2) trained with 15 classes tested with 10 classes, 630 ROIs
EXP_SU	M2s and T72s Tested. Testing Sequestered ROIs vs untrained Unsequestered ROIs.	1) Tested on Sequestered, 120 ROIs 2) Tested Unsequestered, 180 ROIs
EXP_SN	M2s & T72s tested. Measuring performance difference between trained SNs but untrained ROIs against untrained SNs and untrained ROIs	1) ROIs from trained SNs but not the training ROIs, 60 ROIs 2) ROIs from untrained SNs, 180 ROIs

Description of Experiments

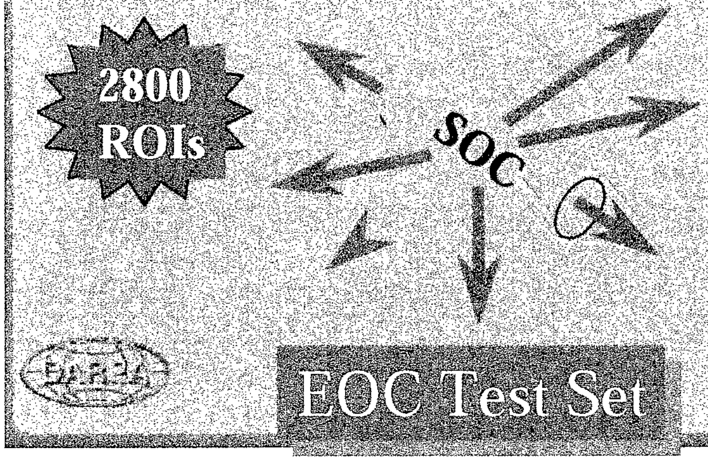
Experiment Name	Experiment Description	Test Bin Descriptions
EXP_ART_RET	T72s tested in their nominal state versus T72s in a combination of low or high articulation coupled with being in 2' or 5' revetments.	1) Nominal, 90 ROIs 2) ART+RET, 140 ROIs
EXP_ART_DEP	T72s, M109s tested. At their nominal state versus in a combination of low or high articulation coupled with being 15Ÿs away from the trained depression angle.	1) Nominal, 180 ROIs 2) ATR+DEP, 290 ROIs
EXP_RET_DEP	T72s tested in their nominal state versus T72s in a combination 2' or 5' revetments coupled with being 15Ÿs away from the trained depression angle.	1) Nominal, 90 ROIs 2) RET+DEP, 110 ROIs
EXP_SQU_COL	T72, M109 and Zil-131s were tested. All training was done with collection 2 vehicles. All testing was done using collection 3 imagery.	1) Broadside squint -100Ÿ-80Ÿs, 120 ROIs 2) Aft squint,-135Ÿ-125Ÿs, 120 ROIs 3) Forward squint,-55Ÿ-45Ÿs, 120 ROIs
EXP_SQU_DEP_COL	T72, M109 and Zil-131s were tested. All training was done with collection 2 vehicles. All testing was done using collection 3 imagery coupled with being 15Ÿs away from the trained depression angle	1) Nominal, 120 ROIs 2) SQU+DEP_COL, 240 ROIs

Confusion Matrix over all EOCs

	TRUCK			APC			MBT		SPG	MML			
	M548	M35	ZIL131	M2	M113	BTR70	T72	M1	M109	SCUD	OTHER	# ROIs	Pid
M548	0.13	0.02	0.10	0.02	0.15	0.02	0.05	0.13	0.07	0.03	0.28	60	0.19
M35	0.02	0.08	0.10	0.02	0.02	0.08	0.03	0.02	0.08	0.03	0.52	60	0.17
ZIL131	0.00	0.00	0.34	0.06	0.12	0.07	0.04	0.03	0.04	0.00	0.29	227	0.48
M2	0.00	0.00	0.11	0.68	0.04	0.05	0.02	0.02	0.05	0.01	0.02	150	0.69
M113	0.00	0.00	0.14	0.00	0.37	0.09	0.00	0.02	0.02	0.00	0.37	57	0.58
BTR70	0.00	0.00	0.14	0.05	0.03	0.56	0.02	0.07	0.05	0.00	0.09	120	0.61
T72	0.00	0.00	0.08	0.08	0.08	0.05	0.55	0.03	0.07	0.00	0.06	719	0.58
M1	0.02	0.00	0.12	0.05	0.03	0.07	0.10	0.53	0.08	0.00	0.00	60	0.53
M109	0.00	0.00	0.09	0.08	0.10	0.09	0.03	0.07	0.52	0.01	0.02	570	0.53

Overall EOC Test Results

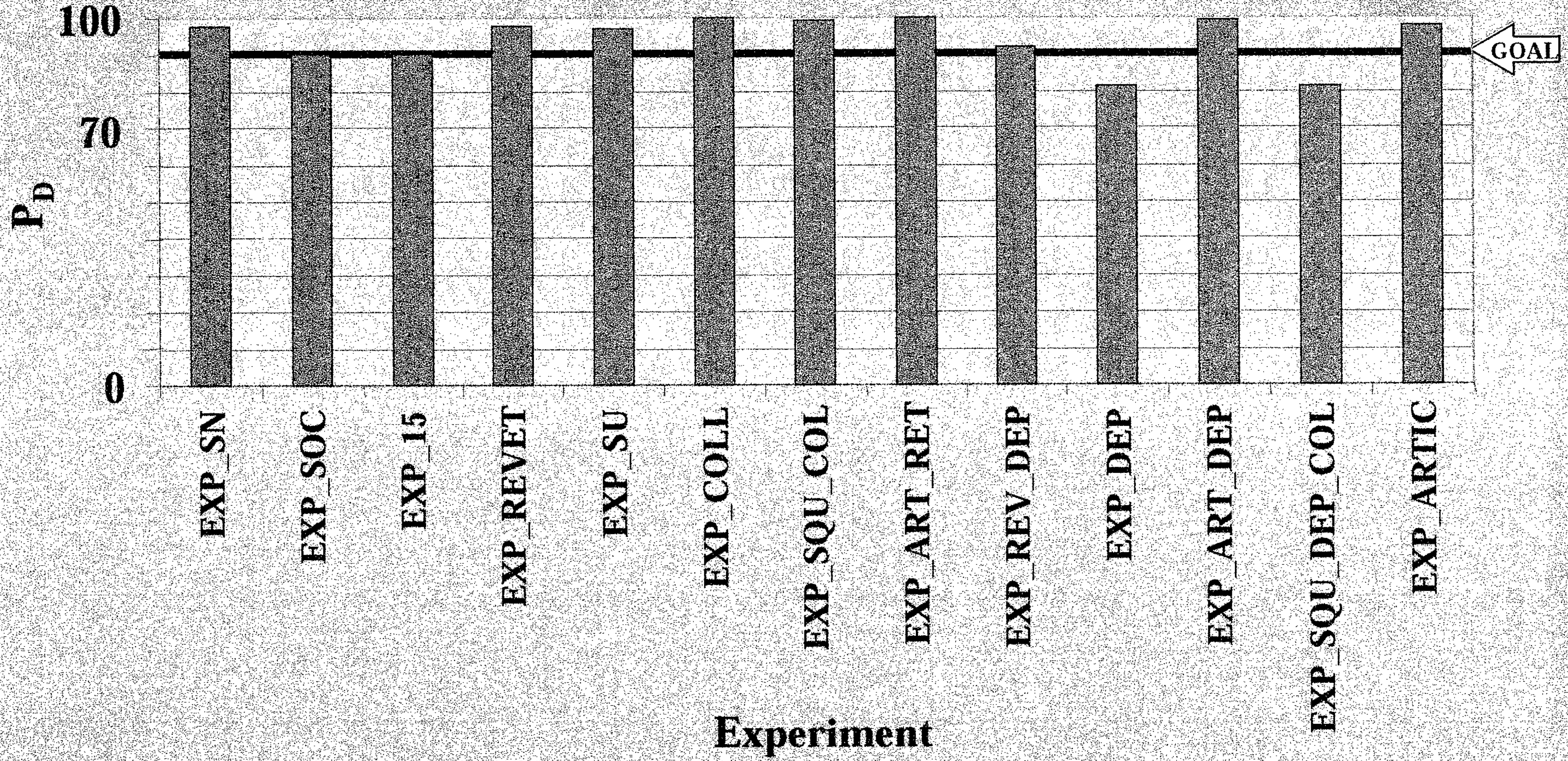
- Test ROIs: The Union of the 12 EOC experiments
- Pd performance for low/no information loss EOC targets
  - M2,BTR70,T72,M1,M109 target types performed above average
  - M548,M35,Zil131,M113 vehicles performed significantly below average
- Pid performance for these same EOCs is signficantly impacted across all types



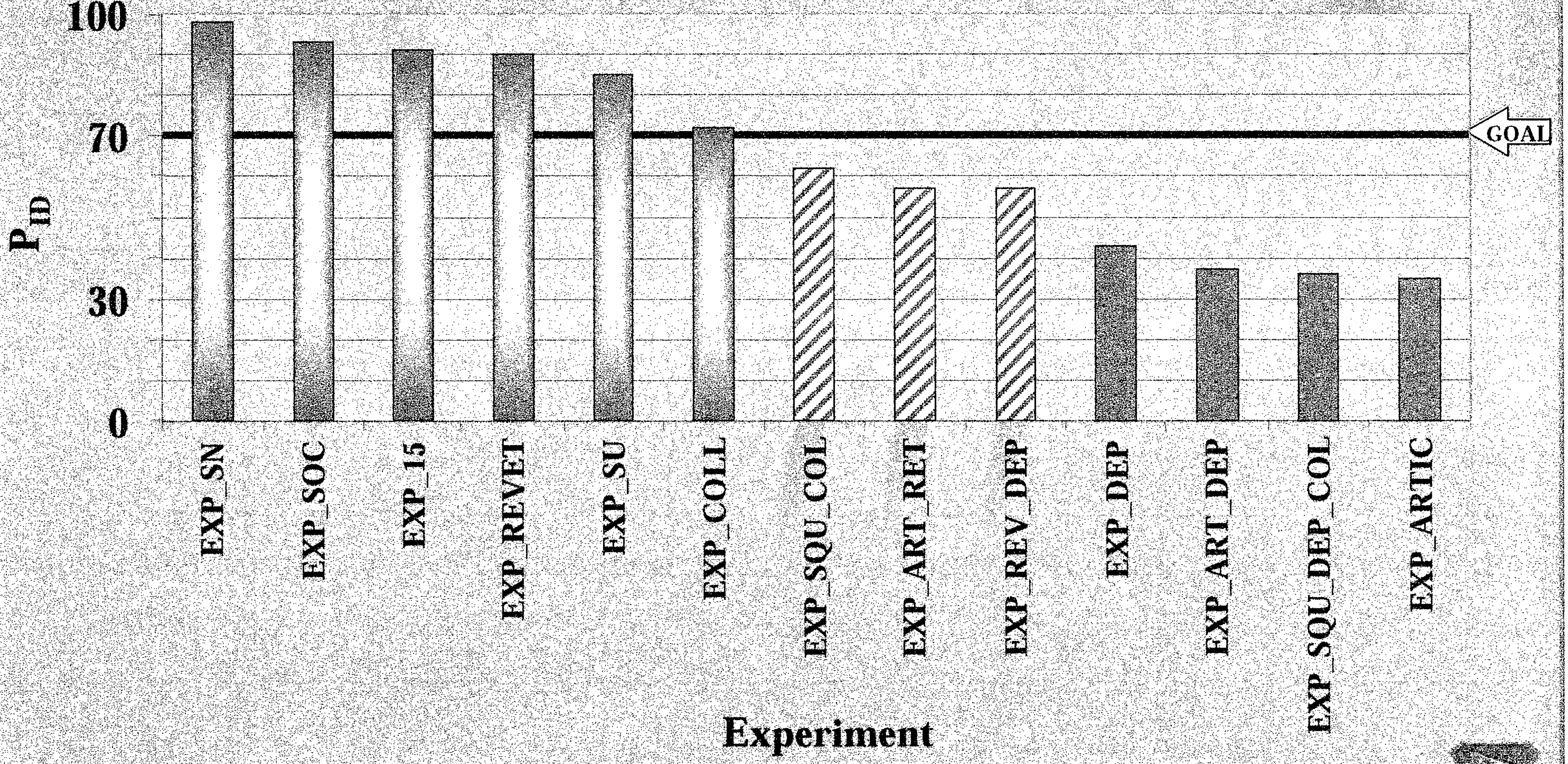
Pd/ROI(S1,EOC) = 0.9 +/- 0.01

Pid/Ty|D(S1,EOC) = 0.50 +/- 0.02

Pd Performance vs Experiment



Pid Performance vs Experiment



EOC Performance Summary

These tests show:

- Pd performance for low/no information loss EOC targets is Insensitive to these EOCs.
- Pid performance for these same EOCs is:
  - Insensitive to: Serial Number, Revetments, Fifteen Class Problem, Sequestration, Collection
  - Impacted by: Squint+Collection, Articulation+Revetment, Revetment+Depression
  - Severely impacted by: Depression, Artic+Depression, Squint+Depression+Collection, Articulation

Web Implementation of the SEI Cleanroom Software Engineering Reference Model

S. Wayne Sherer U.S. Army TACOM LCSEC

Carmen J. Trammell
Software Engineering Technology, Inc.

Introduction

Software engineers at the Tank-automotive and Armaments Command (TACOM) Life Cycle Software Engineering Center (LCSEC) at Picatinny Arsenal, New Jersey ore accustomed to on-line access to the important tools In their environment. In concert with this approach the LCSEC is shifting from a paper based process support system to an on-line process support system. The Picatinny Cleanroom Software Process (i.e., the Picatinny instantiation of the SEI Cleanroom Software Engineering Reference Model) is a web-based interface to process descriptions, work product templates, sample work products, and links to external resources for continuing process improvement. A graphical representation of the Cleanroom process flow provides links to the process descriptions and the corresponding work templates. The process descriptions contain entry criteria, tasks to be performed and exit criteria. The work product templates are downloadable and tailorable to accommodate project needs. This paper relates the Picatinny experience in tailoring the recently published SEI Cleanroom Software Engineering Reference Model for standard use in the TACOM LCSEC.

Cleanroom software engineering is a process for development and certification of high-reliability software systems under statistical quality control. The TACOM LCSEC at Picatinny develops software with high reliability requirements, and is using Cleanroom software engineering in critical product areas.

Cleanroom Software Engineering (CSE)

The management and development team approaches used in Cleanroom Software Engineering (CSE) are consistent with quality management philosophy, i.e., workforce empowerment process focus, and quantitative orientation. it provides for the transition of process technology to the project staff and integrates several proven software engineering practices into one methodology.

CSE consists of a body of practical and theoretically sound engineering principles applied to the activity of software engineering. First a thorough specification phase that results in a multipart specification, including a precise, complete and consistent description of the software part of a system. Software development proceeds from the specification via a step-wise refinement and verification procedure using box-structured design concepts. This phase focuses on defect prevention, effectively eliminating costly error removal phases (i.e., debugging) and produces verifiably correct software. Development of software proceeds in parallel with the development of a usage specification for the software. This usage profile becomes the basis for a statistical test of the software, resulting in a scientific certification of the software quality of the system. Figure 1 provides a high level CSE process model. These three phases; specification, development and certification are performed as independent roles and follow an incremental approach. Increments are kept controllable in size and represent a usable portion of the external behavior to ensure that the teams can maintain intellectual control over their work. The work is performed according to a formal rigorous process.

□ EMBED Powerpoint.Show.7 □□□ Figure 1: CSE Process Model

A high level comparison between the typical development and the CSE philosophy of software development is provided in Table 1. The typical development environment can be characterized by craft based techniques that are highly dependent upon the skills of the individuals involved whereas CSE is an engineering discipline with well-defined processes, rigor and peer review.

Characteristic
Typical Development
CSE

Programs regarded as:
Lines of Instructions.
Correct rule for a function.

Specifications are:
An incomplete description of external behavior and internal design details.
A complete, precise description of external behavior; design details are left for development.

Specifications are transformed into code:
Informally, with debugging to verify code.
Through stepwise refinement and verification using Box Structures.

Failures are:
Expected and accepted.
Unacceptable.

Testing strategy is:
Futile attempt for coverage with little insight into field reliability.
Random sample, based on a usage model, that predicts field reliability.

Intellectual control is:
Rarely achieved.
The intellectual basis for CSE.

Table I: Comparison between Typical Development and CSE

CSE is a formal process that clearly defines the tasks necessary for the engineering effort to progress, the completion conditions for each task, and the control flow that dictates the order of work on each task. Collections of engineering processes also have the same level of formalized control flow and completion conditions. Thus, each engineer, manager or other staff member has well-defined roles and tasks that exist as a part of a larger software process. Project management entails the use of a clearly defined process as the approach to be used to complete the particular project. The intent is to give engineers a clear and understandable road map that they can follow and by which they may track progress towards project completion. Awareness of software process key issue in successfully transferring technology to an organization and to an organization's long-term success applying CSE.

The Software Engineering Institute has defined a Cleanroom Reference Model [Linger 96a] consisting of 14 processes and 20 work products. The Reference Model is intended to be tailored for organization and project contexts. The SEI has mapped the Cleanroom Reference Model to the Key Process Areas of the CMM for Softwaresm [Linger 96b] . The mapping shows that Cleanroom and the SW-CMM ore compatible and complementary.

Cleanroom Engineering Practices

The Cleanroom process is based on an incremental development life cycle. Each increment is a complete iteration of specification, development, and certification activity, producing product and process measures that may be used to determine whether or not the development process is under control.

Mathematically based software development processes are employed to create software that is correct by design. The software specification is represented as a mathematical function, where the domain of all possible input sequences is mapped to the range of correct responses. The design proceeds from the specification in small steps, with team verification at each design step to ensure that each result correctly implements its corresponding specification.

An independent certification team employs statistical usage testing processes. An engineering formalism--a Markov chain--is used to model the population of all possible operational uses, and test cases are randomly generated from the model according to expected (or other) usage probabilities. Statistical testing results in an objective estimate of expected operational reliability in the field.

Field Experience

The Cleanroom process has been demonstrated in numerous software development projects in industry, as well as in NASA and the DoD. Experience has shown substantial improvements over traditional quality and cost [Basili 94, Hausler 94] . Projects have included real-time, embedded, host, distributed, workstation, and client-server systems.

The Picatinny Mortar Fire Control Team, which used Cleanroom over a five-year period in the Improved Mortar Ballistics Computer (IMBC) project, institutionalized Cleanroom as its standard process. The IMBC project experienced substantial gains in productivity (4-to-1) and quality (order of magnitude), and a 20-to-1 return-on-investment for Cleanroom technology introduction [Sherer 96] . A DoD study used results of the Picatinny experience to compare Cleanroom with a number of traditional approaches and found an 80% advantage for Cleanroom in cost savings and a dramatic decline in rework for systems of increasing size [McGibbon 96] . Continuing process improvement activity at Picatinny, such as the implementation of web-based process support, is expected to produce ongoing gains in product quality and process control.

Web-based Process Support

The Tank-automotive and Armaments Command (TACOM) Life Cycle Software Engineering Center (LCSEC) at Picatinny Arsenal, New Jersey, has tailored the recently published SEI Cleanroom Software Engineering Reference Model [Linger 96a] for use as a standard process, and has implemented the tailored process on the LCSEC intranet. The Picatinny Arsenalis web-based implementation of the SEI Cleanroom Software Engineering Reference Model includes process descriptions, work product templates, sample work products, and resources for continuing process improvement.

A graphical representation of the Cleanroom process flow provides links to internal (intranet) and external (Internet) process resources. The work product templates are downloadable and tailorable to accommodate project needs. The SEI Cleanroom Reference Model has a defined mapping to the SW-CMM, and the Picatinny implementation supports all aspects of the Reference Model that are required for compliance with the 18 KPAs in the SW-CMM. The rest of this paper describes the key features of the web-based Cleanroom process implementation.

The Picatinny Mortal Fire Control Team (MFCT) has adapted the Cleanroom Reference Model for local use. The MFCT has used the full set of Cleanroom engineering practices for many years, with process support in the form of training, coaching, and engineering handbooks. The implementation of web-based process support is the most recent step in ongoing efforts to ensure process understanding and adherence. The tailored process flow (shown in Figure 2) is used as the top-level in the MFCT web-based process.

Figure 2. Mortar Fire Control Team Cleanroom Process

Each element in the graph is a link to a lower-level set of process descriptions, work product templates, and resources for continuing process improvement. The Picatinny experience was a key source of user input in the development of the SET Reference Model. As a result only minimal tailoring was required to adapt the Reference Model for MECT use. The following paragraphs include examples of the tailoring activity that was performed to adapt the model for the Picatinny environment and how the model is implemented on the web.

Evolution of the Reference Model

An Increment Specification Process was added to ensure that requirements were fully and correctly specified before proceeding into Increment Design and Usage Modeling. Typical projects are subject to considerable requirements change over their lifecycle and this added step accommodates those changes while still ensuring that the specification is correct and consistent. Another advantage of this added step is that requirements that were not fully understood or specified during the Function Specification step can be deferred to later increments without delaying the project.

Description of Process Steps

The next level of detail for the web implementation expands on each process step. The Increment Planning process step is shown, as an example, in Figure 3. At this level the Purpose of the process step, Process Detail and Work Products are provided. Additional links to Objectives, Participants, Entry Criteria, Tasks, Verification, Measurement, Exit Criteria and Work Product information lead to further process detail.

Figure 3. Increment Planning Process Links

Localization of work products

Work product templates were modified to reflect terminology, roles, relationships, organizational structure, and other circumstances at Picatinny. The work product templates may be downloaded and tailored for project use. Figure 4 shows a portion of the Increment Planning template.

Figure 4. Increment Construction Plan template (excerpt)

Other work product templates were added, modified or replaced. Standard local forms for project resource and status information, for example, replaced the Project Record work product associated with the Project Management process. Figure 5 is an example of one such project status form in use at Picatinny.

Title
Spec
Complete
Spec
Reviewed
Spec
Released
Code
Complete
Code
Reviewed

Ammo Inventory Confirmation
x
x
8/15/96

Ammo Inventory Data
x
x

8/15/96

Ammo Inventory Selection
x
x
8/8/96

Communication Menu
x
x
8/22/96

Communication Setup
x
x
9/22/96

Current Mission Menu

End of Mission Menu
x
x

End of Mission Surveillance
x
x

Fire Data

Figure 5. IMBC Increment 8 Release Information (excerpt)

Summary

This web-based approach to presenting the process details and flow provides easy access to the information needed to implement Cleanroom Software Engineering. It serves as on line help to improve process understanding for the software engineer and provides work product examples and templates that result in consistent implementation. As part of the TACOM LCSEC process effort other project teams are adapting this web-based implementation to support their project processes.

References

[Basili 94] Basili, V.R. and S.E. Green. "Software Process Evolution in the SELî. IEEE Software 11, 7 (July 1994): 58-66.

[Linger 96a] Linger, R.C. and C.J. Trammell. Cleanroom Software Engineering Reference Model. CMU/SEI-96-TR-022, Software Engineering Institute, November 1996.

[Linger 96b] Linger, R.C., M.C. Paulk, and C.J. Trammell. Cleanroom Software Engineering Implementation of the Capability Maturity Model (CMM) for Software. CMU/SEI-96-TR-023, Software Engineering Institute, December 1996.

[McGibbon 96] McGibbon, T. A Business Case for Software Process Improvement. DoD Data and Analysis Center for Software, 1996.

[Sherer 96] Sherer, S.W., A. Kouchakdjian, and P.G. Arnold. "Experience Using Cleanroom Software Engineering". IEEE Software 13, 3 (May 1996) : 69-76.

Authors

S. Wayne Sherer is the chief scientist for the U.S. Army TACOM LCSEC located at Picatinny Arsenal, NJ. He oversees and guides software technology development and transfer into the LCSEC and directs the software acquisition and engineering process improvement efforts. Technology areas include reuse, product-lines, architecture, software support, engineering, quality, process and supporting tools. Mr. Sherer also leads policy, standards, software capability evaluation, contract monitoring and common operating environment efforts. As

ARPA STARS deputy program manager (Army), he was focal point for process technology, product-line methods, and acquisition issues. He has spent over 25 years working on Army and Air Force software intensive weapon systems and is an authorized SEI Lead Evaluator.

TACOM LCSEC
U.S. Army TACOM/ARDEC
Attn: AMSTA-AR-FSF-S, Building 352
Picatinny Arsenal, NJ 07806-5000
Voice: (973) 724-3531 DEN 880-3531
Fax: (973) 724-7850/6828
E-mail: wsherer@pica.army.mil

Carmen Trammell is a software consultant with Software Engineering Technology, Inc. She was formerly Research Assistant Professor and Manager of the Software Quality Research Laboratory in the Department of Computer Science at the University of Tennessee. Dr. Trammell has held technical and management positions in software projects through Oak Ridge National Laboratory, Martin Marietta Energy Systems, and Software Engineering Technology, and currently works with industry and government on software engineering process definition and improvement. She has done software development and testing for the US Army and US Navy, academic teaching on military bases overseas, industrial training for DoD contractors, and R&D under contract to the DoD Software Technology for Adaptable, Reliable Systems (STARS) Program. She received a N.E. in Computer Science in 1991 and a Ph.D. in Psychology in 1981 from the University of Tennessee.

Carmen J. Trammell
Software Engineering Technology, Inc.
5516 Lonas Road. Suite 110
Knoxville, TN, 37909
E-mail: trammell8toolset.com, http: //www.toolset.com
Phone: (423) 450-5151 ext. 240
Fax: (423) 450-9110

AMC

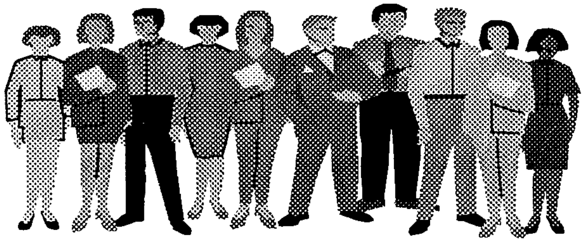
TACOM

Web Implementation of the SEI Cleanroom Software Engineering Reference Model

S. Wayne Sherer
TACOM LCSEC
Chief Scientist
973-724-3531
wsherer@pica.army.mil

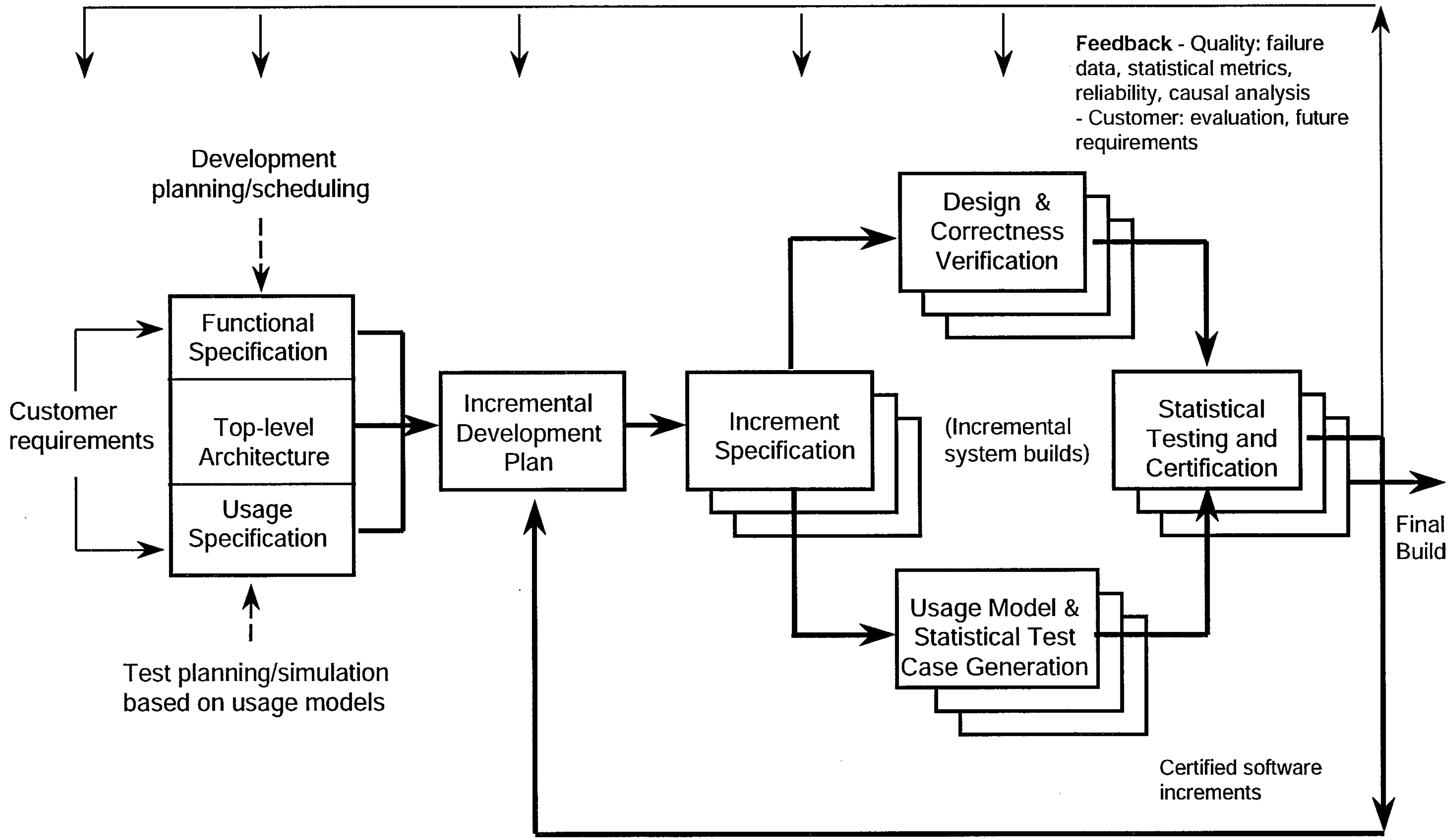
Cleanroom Software Engineering

- Cleanroom developed, used and refined over 15 years at IBM SET
- System developed in multiple increments by small teams using a spiral model approach
- Separate Teams specialized for function
  - Specification
  - Development
  - Certification



- Objective: Zero Failure Software
- Well defined process - Transitionable via training / coaching

Cleanroom Process Model



Feedback - Quality: failure data, statistical metrics, reliability, causal analysis
- Customer: evaluation, future requirements

Development planning/scheduling

Customer requirements

Functional Specification

Top-level Architecture

Usage Specification

Test planning/simulation based on usage models

Incremental Development Plan

Increment Specification

Design & Correctness Verification

(Incremental system builds)

Usage Model & Statistical Test Case Generation

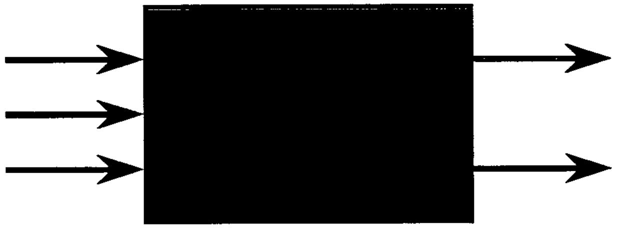
Statistical Testing and Certification

Final Build

Certified software increments

Cleanroom Software Engineering (CSE)

- CSE Projects use Box Structured Design Method
- Black Box Specification for entire system
  - Stimuli
  - Stimuli History
  - Outputs



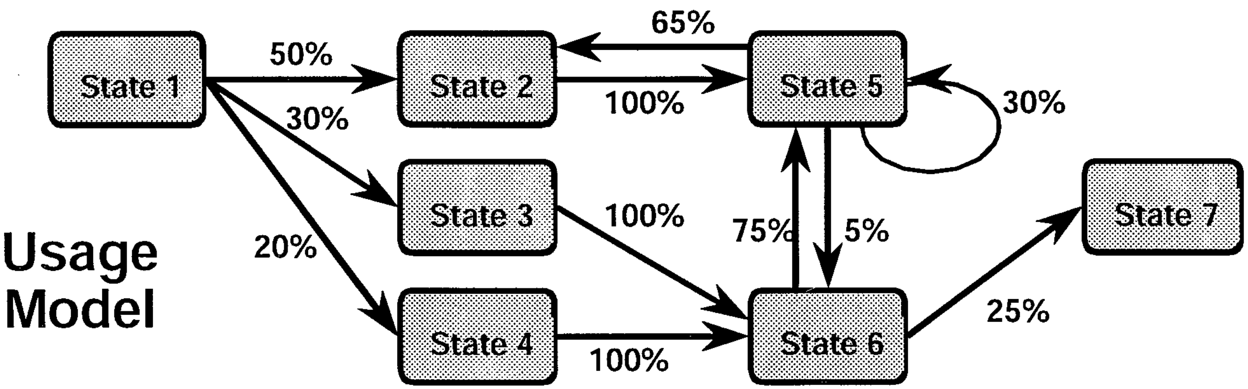
CSE - Development

- Stepwise Refinement: Making small transformations to go from Requirements to Code.
  - Requirements ⟶ A ⟶ B ⟶ C ⟶ ... ⟶ Code
- Functional Verification: Each refinement checked for consistency with previous refinement.
  - Functional Verification C ⟷ B
- This is done to ensure that each transformation is correct, eliminating errors before they are inserted into the code.
- Result: Errors are discovered and eliminated early, resulting in higher quality with higher productivity, by minimizing rework.

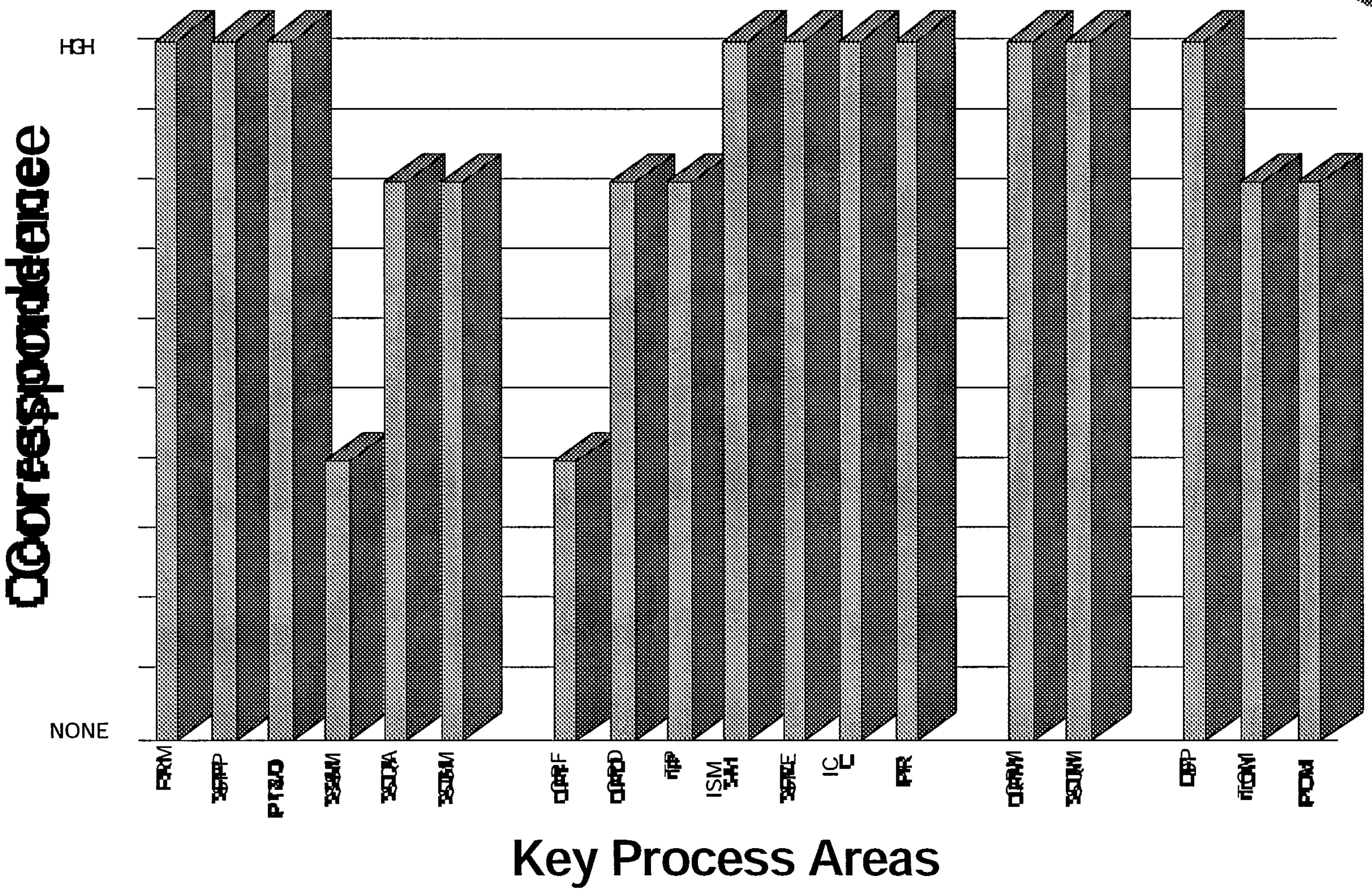
CSE - Certification

- Based upon Usage Model from which Random Sample test cases generate statistical certainty of product reliability and quality
  - Certification performed against product as it is to be used
  - Higher perceived quality from customer perspective
  - Provides a method for deciding test budget

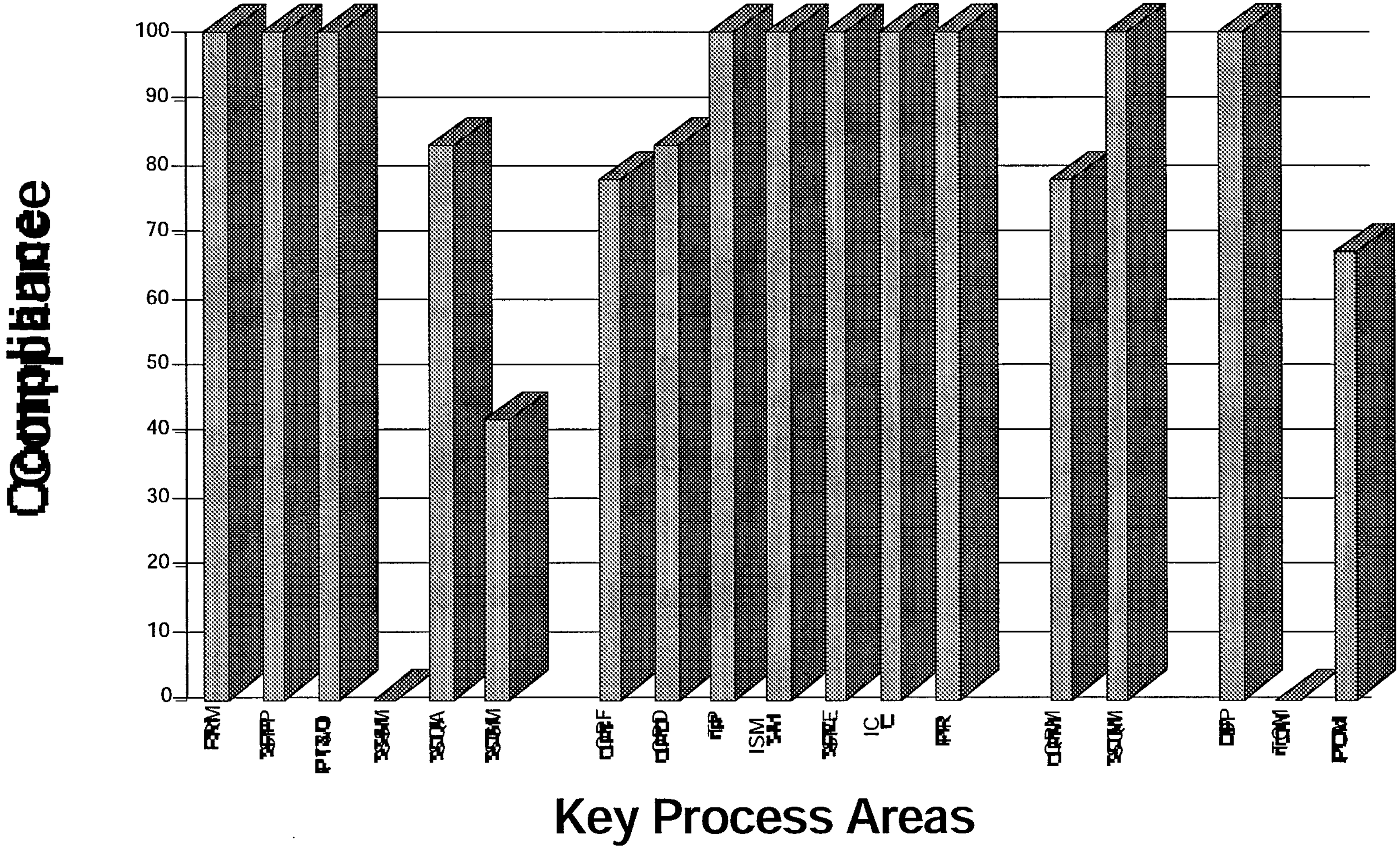
Usage Model



CSE Implementation of SW-CMM

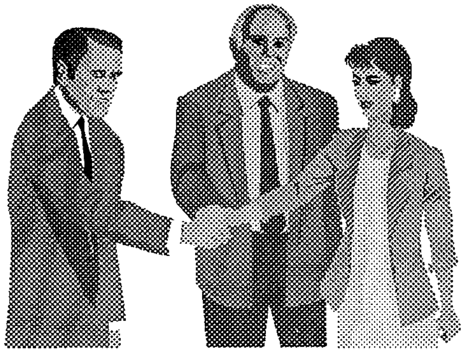


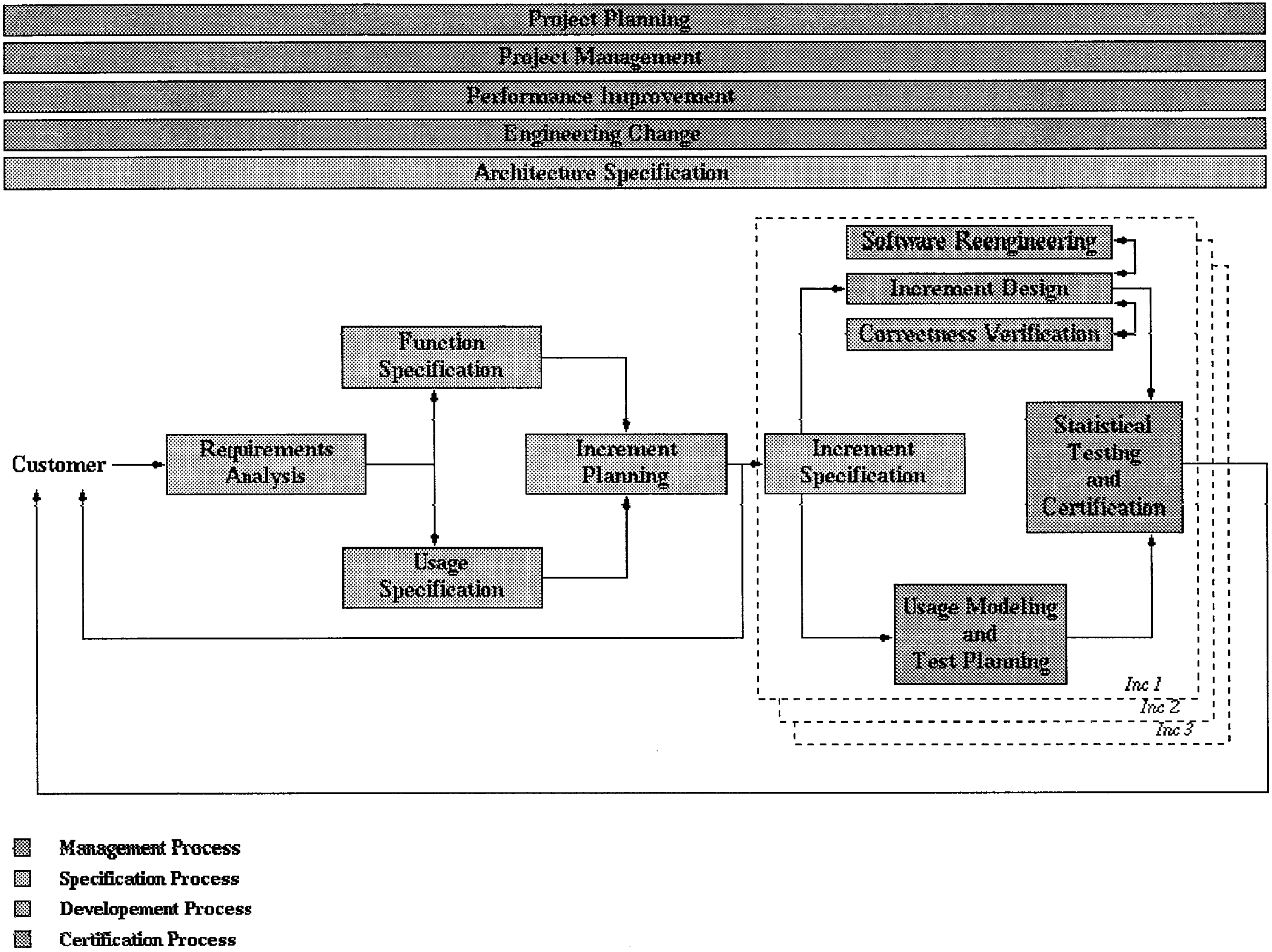
CSE vs SW-CMM Goals



CSE Results for Project at Picatinny Arsenal

- Productivity Change
  - Increments 1 - 7: 559 LOC per staff month (90KLOC of Ada)
    versus
    Baseline of: 121 LOC per staff month
- Quality: Number of Failures
  - Increments 1 - 7: 0.51 Failures per KLOC
    versus
  - Typical Projects: 15-29 Failures per KLOC
  - NASA Space Shuttle: 0.1 Failures per KLOC
- Return on Investment
  - Greater than 20:1
- Planning to use Cleanroom for all future development & re-engineering projects




Project Planning
Project Management
Performance Improvement
Engineering Change
Architecture Specification
Customer
Requirements Analysis
Function Specification
Usage Specification
Increment Planning
Increment Specification
Software Reengineering
Increment Design
Correctness Verification
Statistical Testing and Certification
Usage Modeling and Test Planning
Inc 1
Inc 2
Inc 3
Management Process
Specification Process
Developement Process
Certification Process

APEX: A Model for Avionics Upgrade Planning and Execution

Lt Col Glen T. Logan, USAF
OUSD (A&T)
Open Systems Joint Task Force
2001 N. Beauregard Street
Alexandria, VA 22311-1772
(703) 578-6584
logangt@acq.osd.mil

Charles R. Hurst
Information Spectrum Inc.
5100 Springfield Pike Suite 201
Dayton, OH 45431
(937) 256-9933
hurscr@ispec.com

ABSTRACT: *As our weapon systems age and fewer new systems are under development, the needs of the warfighter for more capable aircraft and systems dictate that these needs be met by modifications to the legacy aircraft the services are currently flying. For aircraft avionics, the needs for technological currency are also compounded by the mandates to operate within the safety boundaries of the National Airspace requirements. The result is a "full plate" for many weapon system managers in establishing a rational plan to acquire and install updated systems in the aircraft they are managing. The authors have defined an approach to this avionics planning process that is described as the APEX (Avionics Planning and Execution) model. The model process shows how to incorporate avionics requirements and modification planning into an effective and integrated plan that considers technical and business case issues. Concepts such as the development of an overall avionics migration strategy, the application of open systems and the use of life cycle cost in the decision process are shown to be key elements of the APEX process. Examples of the application of this planning method to execute on-going programs are provided.*

1.0 INTRODUCTION: The authors evolved the concept of APEX during a series of upgrade planning activities for Air Force legacy aircraft. The APEX embraces the systems engineering and business planning processes and is focused on structuring the information for the customer to use during the modification decision process. The need for an APEX-like process became more apparent with the requirement to install new capabilities on all aircraft operating in the National Airspace. The congressional "mandate" to install Global Positioning System capability on all aircraft by the year 2000 resulted in a number of system managers with a need to incorporate this modification into the other requirements for their respective aircraft programs along with depot maintenance requirements. As the authors assisted several program offices and requirements study projects, it became clear that the approach to avionics upgrade planning was often piecemeal with each modification considered and funded without consideration of the relationship of a particular modification in the context of the overall aircraft capability. Additional factors such as the number of aircraft out of service and costs to "open" the aircraft for modification work were also not factored into the planning process. The end users of the aircraft were lacking an overall view of how the aircraft capability would evolve and thus uncertain of the funding implications since weapon system budgeting had been changed from the supporting commands to the using commands. The APEX then is intended to be a process model that can yield a successful modernization program that meets the user needs and applies current DoD acquisition policies. The concept provides the information for an end user approval process by incorporating cost evaluation as an independent variable and by Open Systems application to ease later upgrades and costs. APEX is an integrated approach to planning that also takes advantage of the Integrated Process Team (IPT) synergism during the development process.

2.0 BACKGROUND: The Defense Budget reductions (*Air Force Magazine*, May 1998) have resulted in few new systems under development and as a consequence have caused more modifications to legacy aircraft to extend service life and add new capabilities, both required for operations and to comply with new National Airspace rules. The budget reductions have also changed the DoD's impact on the marketplace, and together with some acquisition reform considerations, have changed the role of weapon system managers to "buyers" of systems rather than "builders" of systems. Greater use of commercially available systems, components, and parts all the way down to the computer chip level can result in avoiding non-recurring engineering (NRE) costs and permit life cycle cost savings for several reasons. Selection of system elements to integrate into a weapon system modification must carefully evaluate the expected life of the element as well as the likelihood that it will be upgraded by the manufacturer as a product in a family with later products being backward compatible with the original element. This sounds easier than it may be in actual application. Anecdotal stories abound about families of equipment with the classic examples being the PC computer families. The components or elements on which system architectures are based vary greatly in length of service life. Figure 1 shows a recent estimate of the technological life of these various elements. Thus Open Systems application can be a challenging task in developing a systems architecture but an essential ingredient in lowering life cycle costs by permitting both competition and less complicated future upgrades. Application of the open systems concept has been, and remains, a key element of the DoD and component service initiatives under Acquisition Reform.

2.1 Acquisition Reform. In addition to the Open Systems application process being mentored by the DoD's Open Systems Joint Task Force (OS-JTF), other reforms and individual service initiatives are aimed at streamlining the weapon systems acquisition process to reflect a greater reliance on use of available commercially available products and give industry the flexibility to propose designs that integrate these products into proposed avionics modifications. While a full review of the acquisition reform initiatives is beyond the scope of this paper, the impact on the avionics upgrade of legacy systems has been significant. Many of the on-going modification programs have been influenced by the reform initiatives and have caused changes in approach such as stating requirements in terms of expected performance rather than through the application of a number of military specifications and standards in the development contract. Acquisition reform provides for only a minimum number of requirements to be specified by military specifications so that the design trade space will be broad enough for examination of a full range of alternatives by the developing contractor.

2.2 Lightning Bolt Initiatives. As a vehicle to implement the acquisition reforms, the Air Force established the Lightning Bolt initiatives (SAF/AQ Web Pages at http:\\WWW.safaq.hq.af.mil). In addition to other initiatives, one approach is to limit the size of requests for proposal (RFP) by permitting only essential specifications and standards and in general simplifying the solicitation process. The government's goal is to move to insight into contractor's development activities rather that the traditional role of government oversight. A simplified solicitation process, requirements expressed in performance terms, and minimum "how-to" instructions could lead to the conclusion that an analysis/planning study like APEX is not required as the details sometimes worked in pre-RFP studies are now not included in RFP. In the programs where APEX-like processes were applied, the beneficial result was that the end user or customer gained a fundamental appreciation of what the modification entailed, the technical architectures that could satisfy the requirements, and importantly, the anticipated cost. The life cycle cost estimating that is a key element of APEX permits the customer to apply the Cost as An Independent Variable (CAIV) concept as part of the decision process when the program's future is being determined.

2.3 Open Systems Joint Task Force (OS-JTF). Simply stated, the concept of open systems is both a technical and a business approach that applies open (commonly available) standards principally at system interface points. This provides for understood linkages at interface points without requiring that industry open their intellectual property to competitors. With proper system architectural structuring, later modifications or other needs to change the system can be executed with less cost and system impact. The OS-JTF was formed by DoD in November 1994 to develop the policies and processes necessary to embed the application of this concept into the acquisition of weapon systems. The Task Force has approached this challenge on a number of fronts by providing: open system definitions and policies; application guidance and evaluation; training in open systems; sponsoring pilot and demonstration programs to illustrate the concept of open systems; and capturing lessons learned from these Task Force sponsored efforts. Focusing principally on weapons systems electronics, the relationship of open systems with the other DoD system domains was also a concern of the OS-JTF as was the method to implement the policy of using an open system approach in developing the requirements, technical and systems architecture for new or updated systems and business processes. Open Systems becomes a key element in planning and executing weapon system avionics upgrade programs by permitting smaller incremental changes rather than major block upgrades or reworks of the entire aircraft or avionics system.

3.0 APEX PROCESS ELEMENTS: A cursory review of the following elements may lead the reader to conclude that APEX is merely the classic systems engineering approach to developing new or upgraded systems. While the "standard" systems engineering process is an essential ingredient, there are three unique elements of APEX. These are the development of a migration strategy for the aircraft that is time phased to the budgeting process. This shows the strategy for the system projected over the longer term, usually 10 to 15 years. A second unique element is the use of life cycle cost estimates throughout the process to support the system engineering IPT and the customer's program decision and budgeting process. Open Systems application is the third element and the mechanism to effect the modifications and retain the ability to do later upgrades at reasonable cost. APEX can be considered to be the incorporation of new techniques for developing a technical migration strategy, performing CAIV as a recurring process throughout the pre-RFP activities and applying Open Systems. Thus in aggregate, APEX is comprised of no startling new techniques. However, the emphasis on these elements in program planning and the lessons learned from the programs that were the basis for defining the APEX, provide some meaningful insight into the application to legacy programs, and to some degree to new programs. Many of these APEX precepts came from the T-38 Avionics Upgrade Study that subsequently was approved as a formal upgrade program. This program replaces aging avionics, provides for improved reliability and maintainability, adds GPS, and will permit advanced fighter procedures to be taught in the aircraft. For the KC-135 program, some preliminary studies evolved into the development of the on-going PACER CRAG Program that is adding capability and is a first step in modernizing the aircraft cockpit. Other initial studies looked at the since cancelled Non Developmental Airlift Aircraft (NDAA) and the cockpit upgrade to the C-5 aircraft as well as several fighter aircraft studies. Some of the success of getting the T-38 and KC-135 programs through customer approval and budget request start-up can be attributed to the migration strategies that identified the viable alternatives. The life cycle cost estimates were also a key factor that let the customer determine the affordability of an upgrade program. The application of APEX is discussed in relation to the traditional steps in a major avionics modification planning and execution process.

3.1 Requirements Analysis. The requirements analysis process may involve a formalized analysis process such as the Air Force Materiel Command Aeronautical Systems Center's Technical Planning Integrated Process Team (TPIPT) approach that involves a focal point for requirements analysis. Less formal reviews are in IPT technical interchange meetings. The user will translate the early requirements analysis into Mission Needs Statement (MNS) and later into the Operational Requirements Document (ORD). A core need to apply the APEX process is for the requirements baselining process to address all known changes, previously approved modifications, new capability or logistics performance improvements and required changes for safety and National Airspace. The requirement to add Global Positioning System (GPS) capability, for example, necessitated modification planning by each aircraft system manager. In many cases this was not a simple process because of the need to correlate this with other modifications and depot level maintenance actions. The upcoming application of Global Air Navigation and Safety and Global Air Traffic Management (GANS/GATM) to DoD aircraft may present a more complex modification planning challenge for the system managers to address. The authors expect that use of the APEX process can be a valuable contributor to the development of rational plans for individual aircraft types. Both the initial TPIPT process and many Study Managers and Development System Mangers (DSM) activities that precede a formal program establishment involve the aircraft users extensively during the requirement establishment process. In the T-38 program, the users were from both the Operations and Logistics areas of the Air

Education and Training Command (AETC). Both of these areas identified requirements for the upgrade. More importantly, the users continued to be actively involved throughout the entire process of preparing the RFP and completing source selection. The users also participated in support planning and the design review process that occurs between the government and the selected contractor. The user is also key in the next step after requirements have been established: the development of a migration strategy.

3.2 Migration Strategy. Using the laundry list of requirements developed in the requirements definition outlined above, a migration strategy is next developed to describe the phases that will be developed to execute an avionics upgrade plan. A first step is prioritizing the changes to determine the business approach to upgrading the aircraft. If the requirements are extensive and it does not appear to be possible do a single modification that meets all requirements, then multiple phases should be established consistent with the DoD Planning Programming and Budgeting process and other factors such as National Airspace mandates. A top-level phase chart example is shown in Figure 2. After the determination of the number of phases, if more than one is required, the first phase should be depicted by an engineering-prepared functional flow diagram. The IPT then develops notional design alternatives for the system capabilities for that phase. This is accomplished in the form of functional flow charts developed for each of these alternatives. The T-38 study identified only a single phase for upgrade because of substantial cockpit disassembly costs and out of service times for multiple phase upgrades. Within that single upgrade phase an initial three and finally four architectural alternatives were developed for phase 1 evaluation. The KC-135 initial study proposed three phases to complete all projected upgrades and six alternatives were developed for evaluation. Figure 3 shows the migration strategy for the KC-135 at an early stage of evaluation. This may not represent current planning for the aircraft, but served to focus the work that led to the PACER CRAG Program and illustrates the migration of the system technical architecture to show both the removed and the newly installed elements. The requirement for GANS/GATM will require further update of the migration strategy for the aircraft.

3.2.1 Single Phase Migration. A migration strategy with a single phase, while easier to define, also presents many questions that must be effectively answered to establish the formal program. Generally, smaller aircraft may best be upgraded by a single phase to avoid the high costs of cockpit and avionics bay disassembly and reassembly only to have to repeat the task three or four years later. The T-38 program completed several studies that addressed completing the upgrade in more than one phase but the labor cost, lost hardware cost for interim equipment, and user need dates all mitigated against an incremental program. These studies proved very helpful in seeking program establishment and in responding to questions during later year budget reviews.

3.2.2 Multiple Phases. The definition of multiple phases is most likely to be used for larger, older aircraft or systems that require more extensive upgrades than can be accommodated financially in a single large modification. The development of the top-level functionality to be added in each phase also serves as a catalyst to sort the requirements into rational and supportable phases. In the case of the KC-135 aircraft, the support community and the users had been trying unsuccessfully for several years to define a modification program that would add required capability and resolve on-going support problems. These were versions of a then-called Avionics Modernization Program (AMP). Each of the AMP proposals led to the same conclusion: a single upgrade program to update the entire avionics suite was too costly. In addition, individual subsystem managers were competing to have "their" modification funded and installed without a clear perception of an overall plan for the aircraft. Recognizing that funding was a limitation to defining a successful program and that a comprehensive overall plan was needed led the Common Avionics Division of the ASC Subsystems Program office to work with the KC-135 Systems Program Office to define an initial migration strategy that incorporated multiple phases. This included a number cost studies of the alternative ways to upgrade the aircraft in the initial phase. This further helped focus the requirements and frame the architecture that eventually became the on-going PACER CRAG Program. While the GANS/GATM requirements will cause a revision to the strategy for this aircraft, as well as the rest of the DoD aircraft, the use of the migration strategy was the beneficial process that moved the avionics planning for the aircraft from the unsuccessful AMP proposals to an approved upgrade program.

3.2.3 Mapping the Strategy Selected. As shown in the KC-135 migration charts referenced above, the strategy can best be shown in top-level functional flow charts that highlight the change from the current system to the result after any particular phase is completed. The charting method shown in this description of the APEX process was prepared by Dr. Larry Brock of C. S. Draper Laboratory and shows both the addition of new subsystems and components and the items that will be removed/modified during that phase. The functional flow charts and the top-level requirements listing by phase (Figure 2) represent the migration strategy for the aircraft and represent the proposed requirements that need to be satisfied. With this basis, the technical architectures possible for the initial phase can be defined and analyzed. Costs to lead to the start of either contract or in-house design of the actual modification can be estimated.

3.3 Architecture Alternatives. While the development of the alternatives is principally driven by the migration phase being undertaken, the out-year phases must also be considered to avoid any out-year throw-away costs for systems or components that would have a short life. Open Systems consideration is also a key consideration during this stage of defining an avionics upgrade. The benefits of applying Open System interface standards and using commercially available components from a family of products or from products with multiple sources will obviously yield outyear life cycle cost savings and permit easier follow-on modifications during later phases of upgrade. Effective alternatives are a product of extensive market research to determine the performance of available components and systems. When the technical members of the IPT are working with the engineering and marketing staffs of the manufacturers, the initial cost estimates for the expected quantity buy should also be obtained for input to the cost modeling process. The cost estimate for each alternative will be a part of the decision package presented to the IPT and customer.

3.3.1 Specific Alternatives. The market research and engineering evaluation process will identify several viable alternatives to satisfy the functional requirements of the upgrade phase being planned. While some of the technically feasible approaches may be ruled out for various reasons, the mostly likely alternatives should be defined at the component level by preparing lists of the proposed new components. This should include actual or projected performance such as Mean Time Between Failure (MTBF) or Mean Time Between Maintenance Action (MTBMa) and expected item cost. Both the items to be removed and the items to be retained and integrated with the new components should be identified. Functional flow diagrams should be prepared for both the new and retained equipment for the IPT to fully understand the approach. These charts will also be used in the user approval process when the alternatives' performance, costs and notional schedules are presented to the support managers and users. The system level performance expressed as MTBF or MTBMa should be determined from the performance of the individual items proposed for the configuration. For the T-38 AUP three alternatives were defined initially and then a fourth emerged from the IPT evaluation process. The fourth alternative, a hybrid, was a combination of the best features of the other alternatives.

3.3.2 Information Sources.

Direct contact with manufacturers and suppliers will provide the bulk of the information needed to define and cost the alternatives. Other government and industry sources can provide valuable technical, performance and cost information. The Air Force produces and distributes an annual Avionics Planning Baseline (APB) and the Navy produces an Avionics Installation Plan (AIP). These documents describe the avionics systems on legacy aircraft as well as provide other information organized both by aircraft and by avionics systems and components. The APB and the AIP are available to both government and industry. Cost information for components that are currently in use on other aircraft platforms can be obtained from government system managers or program offices. The alternatives can be described at the Line Replaceable Unit (LRU) or component level and provide an appropriate level of detail for cost and technical analysis.

3.4 Life Cycle Cost Estimates. Life Cycle Cost (LCC) estimates are invaluable to the user and supporter in reaching a decision about proceeding with a proposed modification. Constrained defense budgets and the changing military role from a developer of systems to a buyer of systems has established a new paradigm for the role of cost estimating. The Cost as an Independent Variable (CAIV) concept establishes cost as a control factor and encourages the appropriate trade studies during the program definition process. Early in the T-38 Avionics Upgrade study that preceded the formal program, the study manager established a criteria to present likely program costs to the user community as an element of the decision. Target costs for each avionics shipset had been informally provided and were considered as the limiting factors for a program approval. The user also had previous studies completed that estimated the costs for acquiring a new airframe rather that update the 1960s vintage T-38 aircraft. It became obvious at the outset that cost estimates would be a significant element of any decision to pursue a cockpit and associated avionics upgrade. In a similar fashion the KC-135 study that followed the T-38 effort also employed cost estimates to limit some alternatives both from an initial cost and a life cycle cost viewpoint. Thus the authors, who were involved in both efforts, applied a CAIV-like approach before CAIV was defined as a specific formal concept. A flexible cost model can contribute to the effectiveness of the LCC estimates and provide the vehicle to quickly complete the "what-If" excursions that will occur during the review process.

3.4.1 LCC Model. The T-38 study team lead established requirements for use of a cost model that computed operations and support cost based on the predicted equipment performance of the alternatives, basing structure, program phasing, number of aircraft to be modified and the desired maintenance support concept. The Commonality Life Cycle Cost Model (COMMCOST), developed by Information Spectrum and used by both the Navy and Air Force in earlier fighter aircraft studies, was selected based on an earlier study of three different cost models. Figure 4 shows the principal features of the current COMMCOST. The cost estimates became a recurring part of the IPT process for both the T-38 study and the later KC-135 effort and the assumptions and program factors that were used in the model were collectively reviewed by the team. The cost estimates were a team product and not just an accounting "bean count". The engineers collected the costs and the expected performance for model loading during their market research effort to define the alternatives. This information was provided to the model operators, as was the user input for aircraft basing, flying hour projection, program phasing, and maintenance concept. This information permitted the model to project both the acquisition or Investment costs and the Operations and Support costs based on flying rates, MTBFs, repair costs, shipping distances and other factors consistent with the DoD's Cost Analysis Improvement Group (CAIG). For the T-38, as a legacy aircraft, the formal program for avionics upgrade would start in the Engineering and Manufacturing Development (EMD) phase and would be essentially the integration of previously developed or commercially available components. The Research and Development costs were estimated by the engineering team using other comparable programs as an estimating reference. These costs were loaded into the model as pass-through costs. The T-38 study spanned over 18 months and considered a number of alternatives and programmatic variables while the KC-135 was a short, less detailed, two month long look at the alternatives for the first phase of an upgrade. The cost estimates were strong supporting material for the decision process and yielded several lessons learned for incorporating a CAIV process into a development program.

3.4.2 LCC Lessons Learned. The use of an IPT process that had costing inputs from several specialties and made cost an integral part of the study resulted in a sense of "ownership" of the estimates. This helped the decision review process proceed on schedule. As the analysis of alternatives proceeded, the impacts on LCC of various components was considered in relation to other configurations that provided the same functionality. By configuring the model only to compare portions of the overall system, the relative costs could be determined and used for a cost comparison rather than developing an overall LCC estimate. There are differences between a life cycle cost estimate and the PPBS ground rules for a Program Objective Memorandum (POM) input. However, These differences should be understood and the cost estimates will need to be converted into a budget request if the user decides to pursue the modification. A study team should expect a detailed and comprehensive review by the financial staff when a program decision is made and before the program funding request is submitted in the POM. In the case of the T-38, the financial review confirmed the validity of the cost estimate and the principal activity was to convert the cost estimate to budget "rules".

3.5 Analysis of Alternatives. The analysis of alternatives represents the IPT action to review both the technical and cost features and shortcomings of the various design approaches. This process could also result in the development of additional alternatives. The integration of the proposed systems in terms of space and environmental factors are compared as are the fly-away cost and the projected operation and support cost estimates. A rating or scoring system for this process would at first appear to be of value in the comparison, but several attempts to develop such a metric were unsuccessful and the end result was a subjective review of individual factors among the alternatives.

3.5.1 Technical Analysis. In the T-38 Avionics Upgrade Study, three architectures were proposed as the most likely choices and these alternatives were then developed using the methods outlined above; market research and cost estimation. Early in the analysis of alternatives it became obvious that there was a viable fourth or hybrid alternative. The original alternatives were architectures based on MIL-SPEC equipment, commercial components or modular avionics. Each of these alternatives had some drawbacks that led to the hybrid alternative. The MIL-SPEC approach required components that were not available as military equipment. This included systems such as a Traffic Collision Avoidance System (TCAS) and others. The commercial alternative did not represent a viable alternative because some user requirements could not be met with available commercial equipment. The Head-Up-Display systems projected to be available at the time of the study would not have the capability to be effective when used in training for future fighter pilots. The modular architecture showed many positive and desirable features but required development, and accordingly, would have increased cost and to some degree increased the risks in a formal program. During the definition of the three alternatives the "shortages" of systems were overcome by using other approaches for systems such as the HUD or TCAS, yet each alternative had some drawbacks remaining. The hybrid was a selection of the best features of the original alternatives and became part of the evaluation and cost estimating process.

3.5.2 Cost Estimate Analysis. For each of the alternatives the production quantity cost estimates provided to the engineers by equipment suppliers during the market research were used in the cost model to calculate the investment or production costs based on an assumed number of aircraft and program phasing. The O&S cost estimate was calculated by the model based on the flying hours, basing structure, Mean Time Between Maintenance and other factors such as transportation costs for off-base component repair under a two-level maintenance. Since the proposed modification was to be basically the integration of existing capability into a legacy aircraft, the engineering cost estimates were developed from the several recent programs accomplishing comparable aircraft upgrades. Using the approach of listing the complete configuration for each alternative at the LRU level permitted the calculation of fly-away costs as well as total program cost estimates. These turned out to be important factors in the user decision process. The availability of cost estimates based on performance, configuration and program phasing also permitted the user to request and receive the results of many "what-ifs" in the process of standing up a formal program. Later evaluation of the cost estimates confirmed the approach of getting industry quotes versus using cost models designed to estimate component costs based on weight, complexity or other technical factors. Another factor the user found beneficial in the decision process was the calculation of current support costs or baseline costs for the unmodified aircraft using the same cost factors as was used in the alternatives costing. The cost estimates were clearly an independent variable in reaching a program decision. Comparisons showed that electing to not pursue the program would result in approximately the same life cycle costs over 20 plus years but would not gain the ability to teach advanced training tasks and stop the declining avionics reliability. See Figure 5. The authors presented this chart to the using command's project managers and this became a factor in presenting the concept to senior management for program approval. The issue of requiring a large up-front expenditure to gain the overall life cycle advantage was a key consideration, but the declining performance and usability of the aircraft as a pilot training platform was a deciding factor. The cost estimates confirmed that the upgrade was an affordable program.

3.6 Program Approval. The results of the analysis of alternatives provide the vital data for the approval of an avionics upgrade program. A major pacing item that served to move for an early decision on the T-38 program was the requirement to have Global Positioning System (GPS) installed on the aircraft by the year 2000. The method to add this capability and the decision to integrate the GPS into the navigation solution display for the pilots resulted in several additional small studies to look at options for adding GPS first or including this as part of the upgrade program for the entire avionics suite. All DoD aircraft basically faced this timing dilemma in adding this capability. While this was a difficult process for several programs, the T-38 avionics upgrade study provided the technical and cost impact information to develop a rational and defensible plan for the aircraft. Programs that did not have a migration strategy for the aircraft being managed were in the position of resolving production conflicts with individual modifications and maintenance requirements. The authors are convinced that changes to National Airspace requirements will pace aircraft avionics changes and will occur again in the next several years as the GANS/GATM modifications and timing are solidified. Program approvals will likely be an iterative process, but demonstration that market research has yielded viable alternatives and that adequate cost estimates have been done with the involvement of the user community present a strong argument for the modification program. Both the supporting commands and the using commands will have a role in the approval process. This will usually require strategic and tactical program roundtables and the "conversion" of the cost estimate into a budget request by factoring in lead times and any changes from the financial management review of the estimate. Program approval should also fully consider the future expandability and flexibility that can be attained in an upgraded system. As a buyer of systems, the Open Systems approach advocated by DoD through the OS-JTF will provide outyear expandability and flexibility, avoid obsolescence and provide life cycle cost savings. Pilot and demonstration programs that the OSJTF has helped sponsor with the services and industry have shown the potential of using commercially available components and software in our military systems. Open Systems have become a favorite word choice in many requirements documents, but unless fully considered during the decision process will not provide the ability to attain the outyear benefits of the use of open, commercial systems.

3.7 Solicitations for Avionics Upgrades. For post-production legacy aircraft, many avionics upgrades will be of the scope that requires a competitive solicitation and source selection process. Acquisition reform has changed the way requirements are stated in the Requests for Proposal (RFP). The old way of specifying requirements was to list numerous military specifications and standards to be used in the design process. Now the approach is to state the required performance for the system being acquired using very few specifications, or at best, using industry recognized open standards. Reviews of government requirements documents will likely show a requirement for Open Systems and the industry response to RFPs will propose that the design that will be provided will be based on Open Systems application concepts. The full incorporation of the Open Systems process requires greater emphasis on evaluating how the proposed design provides for expandability, flexibility, and affordability. The RFP should clearly indicate an intent to use these factors as major elements of the source selection process.

3.8 Source Selection. During the source selection process, the proposals should be carefully evaluated based on the degree of openness provided by the design. Open Systems designs could initially cost more but provide the framework for later modifications or upgrades. However, a clear ability to minimize later hardware or software costs can be the basis for a best value source selection that recognizes the innovation of a life cycle approach rather than one-time cost as the criteria.

3.9 Upgrade Program Execution. As an avionics upgrade program moves to the design stage, the openness of the design and the performance of the support factors such as MTBF or MTBMa should remain up-front considerations in approving the various design review levels that have been established for the program. With the acquisition reform goal to change government oversight to government insight this may represent a paradigm shift for the industry/government teams in the execution of a large avionics upgrade program.

4.0 APEX OPERATING PRINCIPLES. The APEX approach can be accommodated best by the use of the operating principles that the authors have observed during a number of avionics planning efforts. These are, in effect, the lessons learned in how to apply the concept to better develop and manage the avionics capability and currency for an aircraft type.

4.1 Use an Integrated Product Team Approach. Both government and industry are quick to indicate that an IPT approach is being used, but to make the process more than a series of meetings requires an open forum where viewpoints and approaches cut traditional functional boundaries and become task oriented. For upgrades that occur during a production run, this may represent the on-going discussions between the prime contractor and the government program office. For legacy programs that are out of production, the effort will be mainly a government process with industry interaction through the market research process and through pre-solicitation industry days and other forums. For both the T-38 and KC-135 studies, the definition of alternatives, collection of performance and cost data, and the preparation of decision information was undertaken by a team that generally ignored the traditional functional boundaries. The collection of the quantity cost estimates by the study engineers to insure that costs of a proposed unit are consistent with required LRU performance is an example of a task orientation rather than the traditional functional approach.

4.1.1 IPT Operation. Early in the definition and development of the architectures that will be evaluated as alternatives, the tendency of the IPT can become a process of trying to "design around a conference table". The IPT should appropriately assign tasks for presentation to the group and hold frequent meetings to check results and progress.

4.1.2 User Involvement. The direct and continuous involvement of the aircraft users and supporters in the IPT is necessary to retain a constant focus on the requirements the upgrade is working to provide. In the T-38 study, both the operational users and the logistics supporters were involved from the start of the IPT, as the upgrade is to provide improvements in both areas.

4.2 Migration Strategy is Key. The migration strategy provides a method to set the overall framework for incorporating on-going modifications, National Airspace requirements, and new operational and supportability requirements into a cohesive and executable program. The migration strategy integrates the known requirements, facilitates the definition of upgrade phases, and permits consideration of open system approaches and affordable upgrade programs consistent with available user funding. A migration strategy is useful in getting user buy-in for the upgrade approach before the full study resources are applied to alternative definitions and cost estimations. New alternatives for possible architecture alternatives can and most likely will occur, but the migration strategy works from a usually consistent set of user requirements. This requirement set can also benefit the user's formal requirements process through the Mission Needs Statement (MNS) or Operational Requirements Document (ORD) The migration strategy should be based on a an executable notional schedule which will also serve as the program shell for the cost estimate process. Several past efforts to develop avionics roadmaps have been attempted but these became listings of individual modifications, usually at the LRU level. These roadmaps did not integrate the requirements into a strategy that considered all requirements and generally were designed to show modification funding status and not to serve as a tool to plan the future of the avionics system for the respective aircraft.

4.3 Cost as An Independent Variable. The use of cost estimates for the evaluation of alternatives permits to users to select the desired alternative and address the affordability of both the strategy and the upgrade phase being considered. The cost estimates developed for the T-38 were fairly detailed and showed the impact on life cycle costs as well as the near-term costs for the first phase of the strategy. This cost information served to provide a source of information to answer many questions during the formal decision process and became the basis of the program budget. During a timeframe where budgets were tightening, the ability of the program to proceed from study to a new start avionics upgrade program is attributable, at least in part, to the cost information available to the program managers. Use of a flexible cost model, such as COMMCOST, that can be structured to closely match the proposed program and run excursions and sensitivities is a major help in the approval process and the program stand-up. The cost estimation process can also provide unexpected benefits in an APEX process. In one program, the support agency was developing a modification to replace a particular avionics component. Model runs showed that the modification would actually degrade LCC, as the component proposed to be replaced was not the factor in the performance degradation. Associated components were the problem driving the costs and this was demonstrated to program management. The result was the deletion of the modification in favor of a different solution embedded in a phase of a migration strategy.

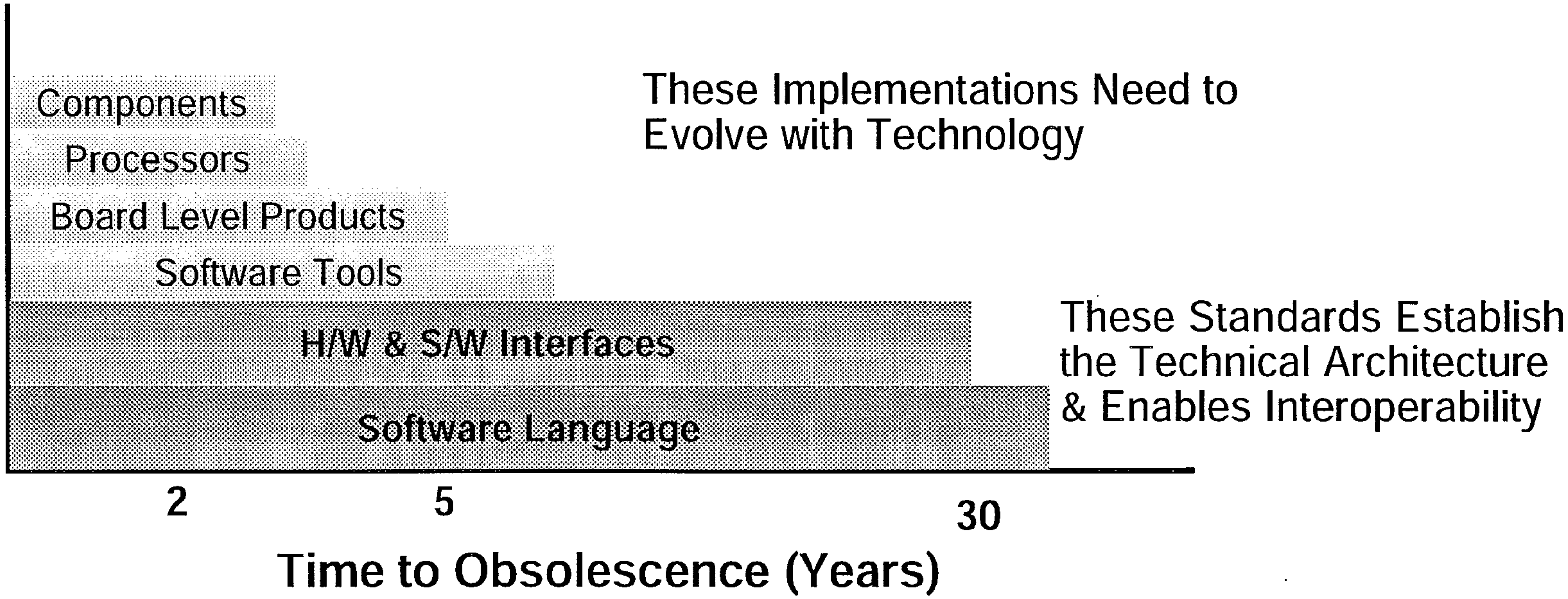
4.4 Open Systems and Migration Strategy. While an aircraft avionics migration strategy could be based on products with closed, propriety interface standards products, this does not permit later modifications at least cost nor take advantage of the commercial upgrade of families of equipment. The addition of a new avionics component through a modification only to have that component removed for scrap a few years later because of a system interface requirement can be avoided by keeping an Open Systems approach throughout the planning process. As discussed above, Open Systems concepts and approaches should be clearly stated and evaluated during the solicitation and source selection. Open Systems are also a factor that should be considered during the market research conducted to define alternatives. Selection of technologies that are predicted to become obsolete or do not have an open interface defined can result in the throw away costs when new capability is added later. The degree of openness of a proposed system is difficult to measure and not all architectures can be fully open. The goal is not Open Systems as an end unto itself but the ability to affordably upgrade and sustain fielded systems. Open Systems application guidance is currently being emphasized by in-work changes to the Joint Aeronautical Commanders Group's (JACG) Performance Based Business Environment (PBBE).

5.0 APEX PROGRAMS. Since the APEX process as defined in this paper has evolved over a six year time period while working the referenced programs and other similar avionics studies, no single upgrade program has applied all of the APEX principles. Notably, the T-38 Avionics Upgrade Study that transitioned into the T-38 Avionics Upgrade Program (AUP) is now approaching first flight. The T-38 study provided a large measure of the insights from which the APEX was defined. The KC-135 avionics study was an abbreviated study, but provided an opportunity to understand the ramifications of an upgrade that would require multiple phases to accomplish. The authors are currently working with avionics modernization for an aircraft still in production for which managers are coming to grips with the retrofit needs and the pending GATM requirements. We suspect that the APEX will further evolve during this effort. The authors do not claim that APEX represents all new concepts. However, it is an approach to avionics planning and execution that emphasizes use of a migration strategy, life cycle CAIV cost estimation and the application of Open Systems concepts.

6.0 ACQUISITION REFORM. The on-going acquisition reform process could be viewed as presenting a paradox with a proposition that advocates detailed analysis of the architectural alternatives, cost analysis and Opens Systems concepts. If the government will only specify performance requirements and industry will propose the design solution then what is the purpose of the up-front and rather detailed study? The authors do not share this view. While there is a level of frustration in stepping through the design process a second time, the market research sets the framework for understanding the range of possible solutions. The cost estimates were invaluable in the program approval and in the establishment of a budget for the program standup. The Open Systems approach can provide the "building codes" that let us upgrade avionics rationally, affordably and within the timeframes of warfighter needs. While acquisition reform has changed many paradigms and concepts, APEX does not appear to be at odds with this new way of doing business.

7.0 SUMMARY. The planning and execution of avionics upgrades for legacy aircraft can be a challenging process for system and program managers as they work to incorporate warfighter needs into the aircraft systems in a timely and affordable manner. Over the past six years the authors have been involved in a number of avionics planning and execution programs and have defined the APEX model process from that experience. The APEX suggests that an overall migration strategy for the avionics upgrades, greater use of Life Cycle Cost estimation and the application of Open Systems concepts are the three principal legs of the process. The authors are more than willing to discuss the APEX process and share their enthusiasm for improving the avionics planning process, a core ingredient in mission execution.

Technology Independent Technical Architecture Supports Evolving Implementation



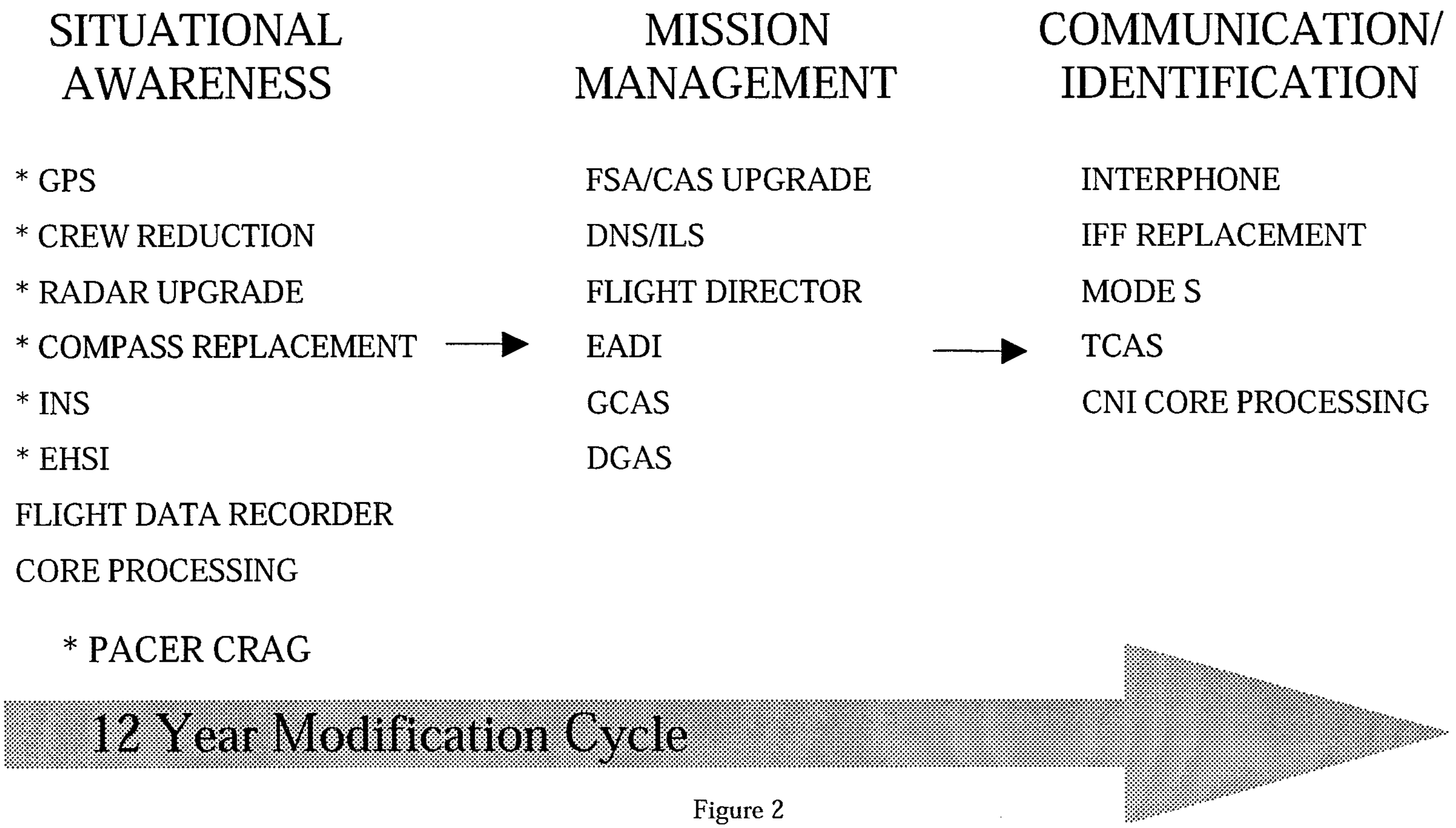
These Implementations Need to Evolve with Technology

These Standards Establish the Technical Architecture & Enables Interoperability

Setting Foundation in Long Lived Standards Allows Implementations to Evolve

Figure 1

KC-135 AVIONICS UPGRADE
(Notional Phasing)



SITUATIONAL AWARENESS

- * GPS
- * CREW REDUCTION
- * RADAR UPGRADE
- * COMPASS REPLACEMENT →
- * INS
- * EHSI
- FLIGHT DATA RECORDER
- CORE PROCESSING

MISSION MANAGEMENT

- FSA/CAS UPGRADE
- DNS/ILS
- FLIGHT DIRECTOR
- EADI →
- GCAS
- DGAS

COMMUNICATION/ IDENTIFICATION

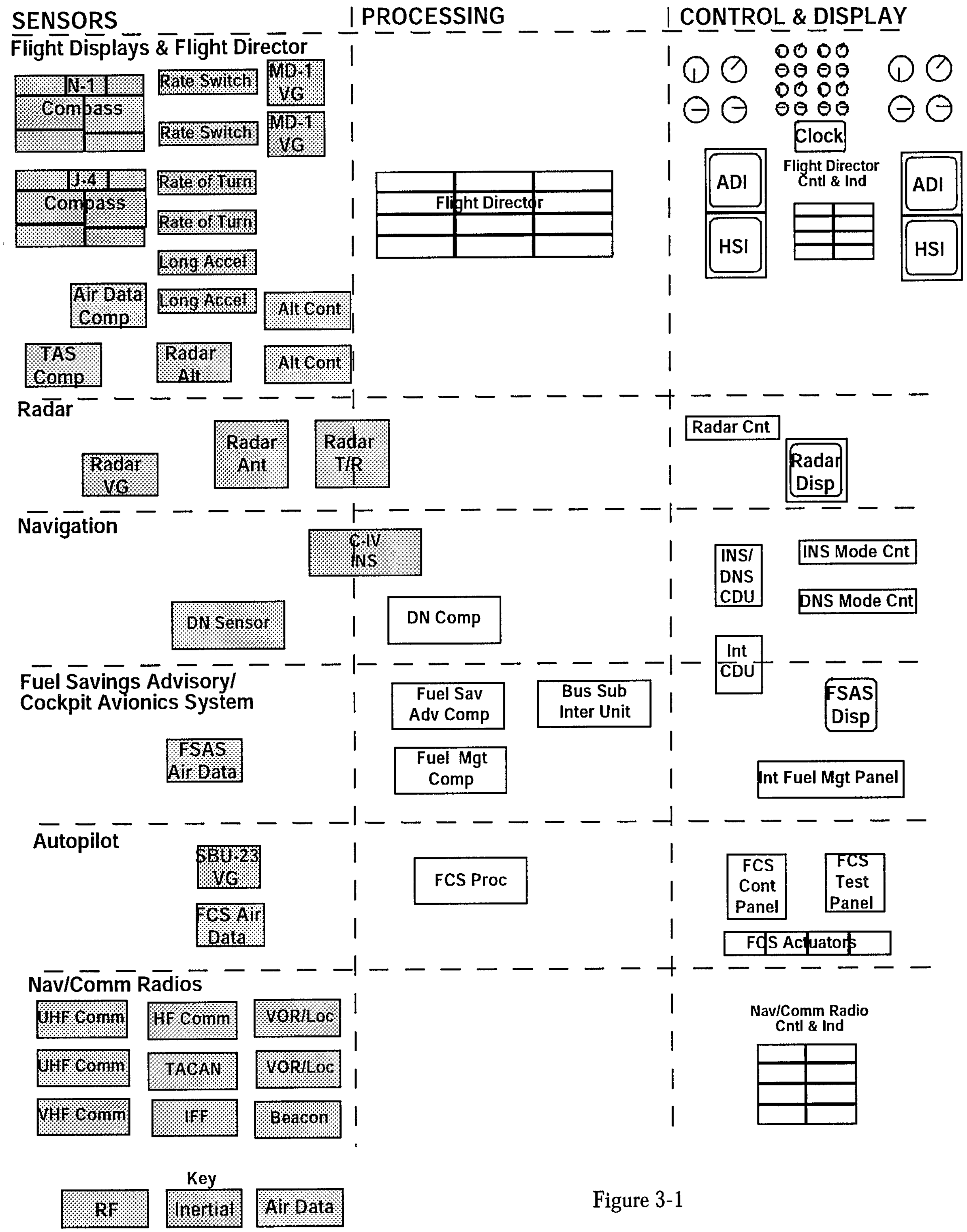
- INTERPHONE
- IFF REPLACEMENT
- MODE S
- TCAS
- CNI CORE PROCESSING

* PACER CRAG

12 Year Modification Cycle

Figure 2

Current KC-135



SENSORS | PROCESSING | CONTROL & DISPLAY

Flight Displays & Flight Director

N-1 Compass

Rate Switch

MD-1 VG

Rate Switch

MD-1 VG

J-4 Compass

Rate of Turn

Rate of Turn

Long Accel

Air Data Comp

Long Accel

Alt Cont

TAS Comp

Radar Alt

Alt Cont

Flight Director

Clock

Flight Director Cntl & Ind

ADI

HSI

ADI

HSI

Radar

Radar VG

Radar Ant

Radar T/R

Radar Cnt

Radar Disp

Navigation

C-IV INS

DN Sensor

DN Comp

INS/ DNS CDU

INS Mode Cnt

DNS Mode Cnt

Int CDU

Fuel Savings Advisory/ Cockpit Avionics System

FSAS Air Data

Fuel Sav Adv Comp

Bus Sub Inter Unit

Fuel Mgt Comp

FSAS Disp

Int Fuel Mgt Panel

Autopilot

SBU-23 VG

FCS Air Data

FCS Proc

FCS Cont Panel

FCS Test Panel

FCS Actuators

Nav/Comm Radios

UHF Comm

HF Comm

VOR/Loc

UHF Comm

TACAN

VOR/Loc

VHF Comm

IFF

Beacon

Nav/Comm Radio Cntl & Ind

Key

RF

Inertial

Air Data

Figure 3-1

Notional Upgrade Phase 1

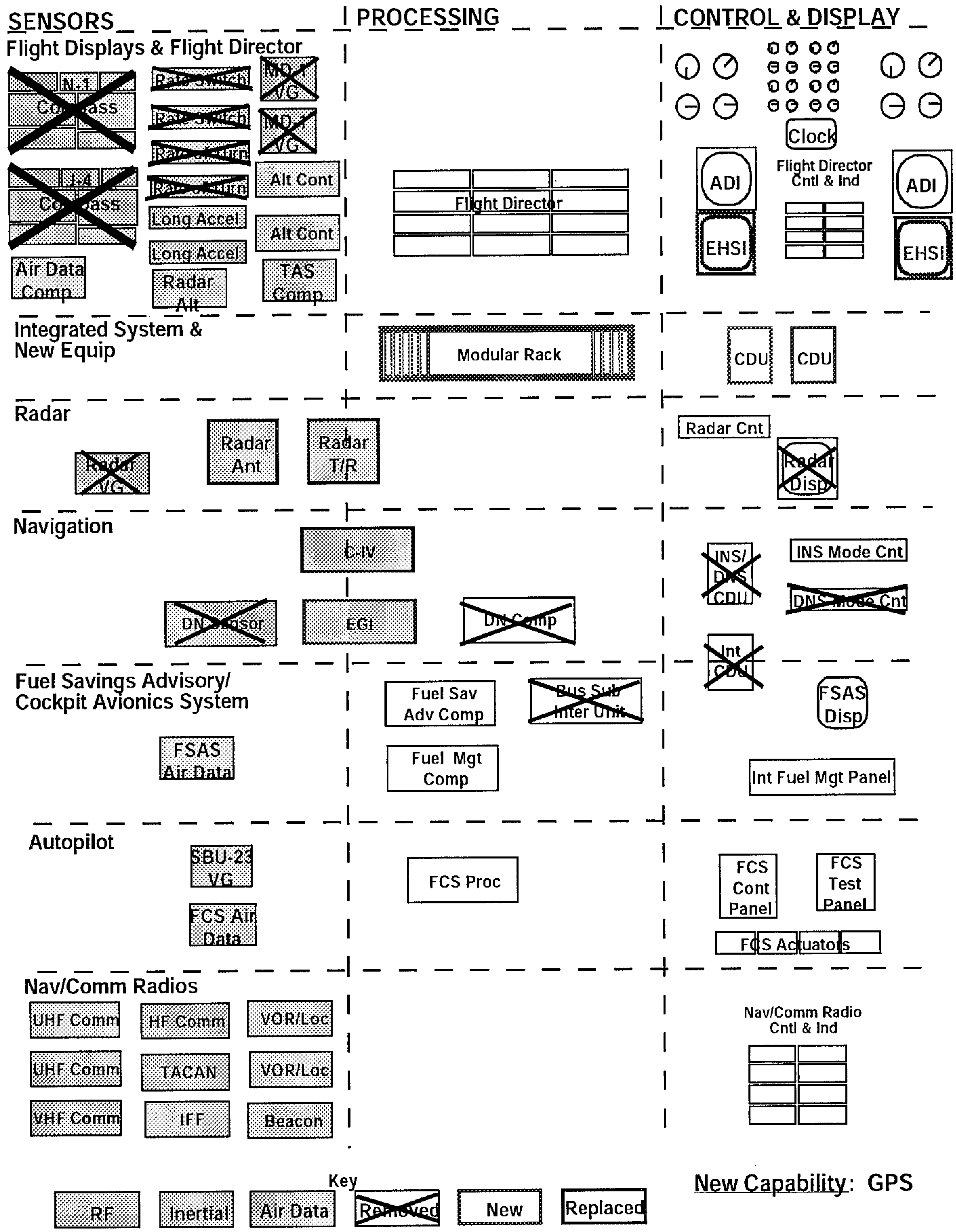


Figure 3-2

Notional Upgrade Phase 2

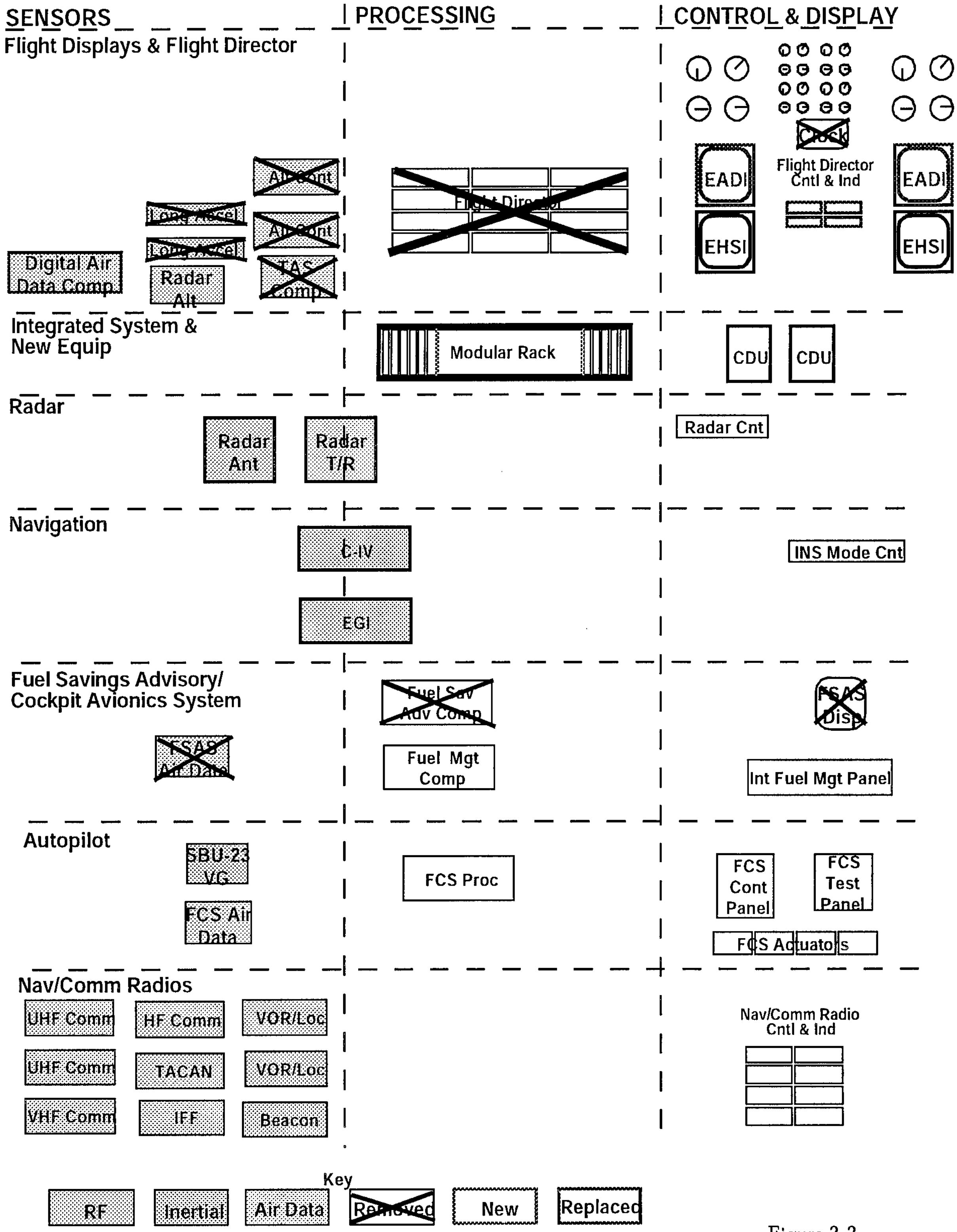


Figure 3-3

Commonality Life Cycle Cost Model (COMMCOST)

- DEVELOPED FOR NAVAIR/JOINT INTEGRATED AVIONICS WORKING GROUP (JIAWG).
- DESIGNED AS AN AVIONICS ANALYSIS COST TOOL.
- FLEXIBLE, ANALYST TOOL: MULTIPLE WAYS TO CALCULATE SOME COSTS.
- CAIG CRITERIA IS USED, LEARNING CURVES, DOD INFLATION FACTORS.
- CAN BE AN ESSENTIAL PART OF AN INTEGRATED SYSTEMS ENGINEERING PROCESS.
- BROAD PARAMETERS FOR CASE VARIABLES.

Figure 4

T-38 AVIONICS UPGRADE
LIFE CYCLE COST COMPARISON
490 AIRCRAFT

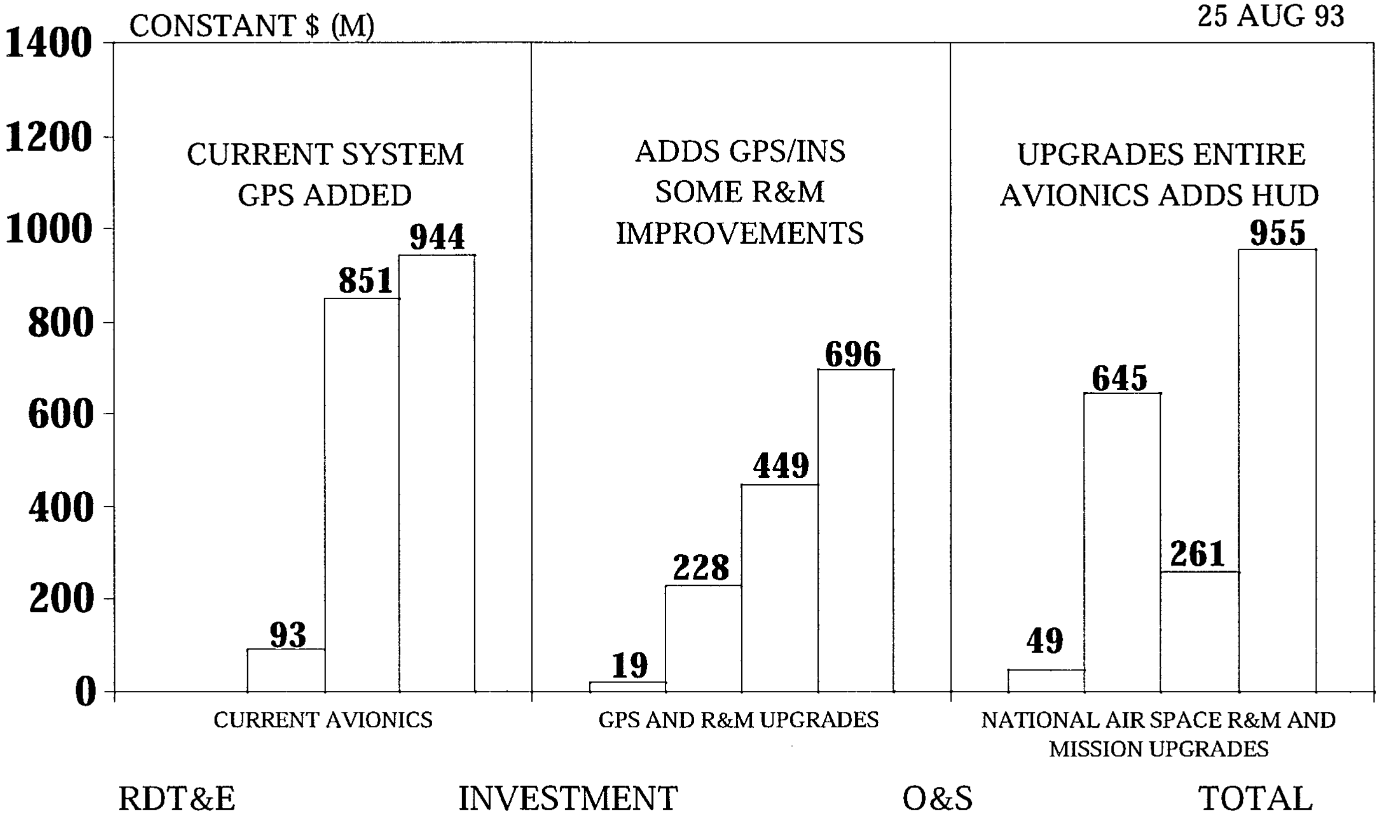


Figure 5

*W*ORKSTATION for *E*LECTROMAGNETIC *R*ADIATION *M*ODELING & *S*IMULATION

CLIFF ALLEN

COMPUTER SCIENCE AND APPLICATIONS, INC.

WERMS

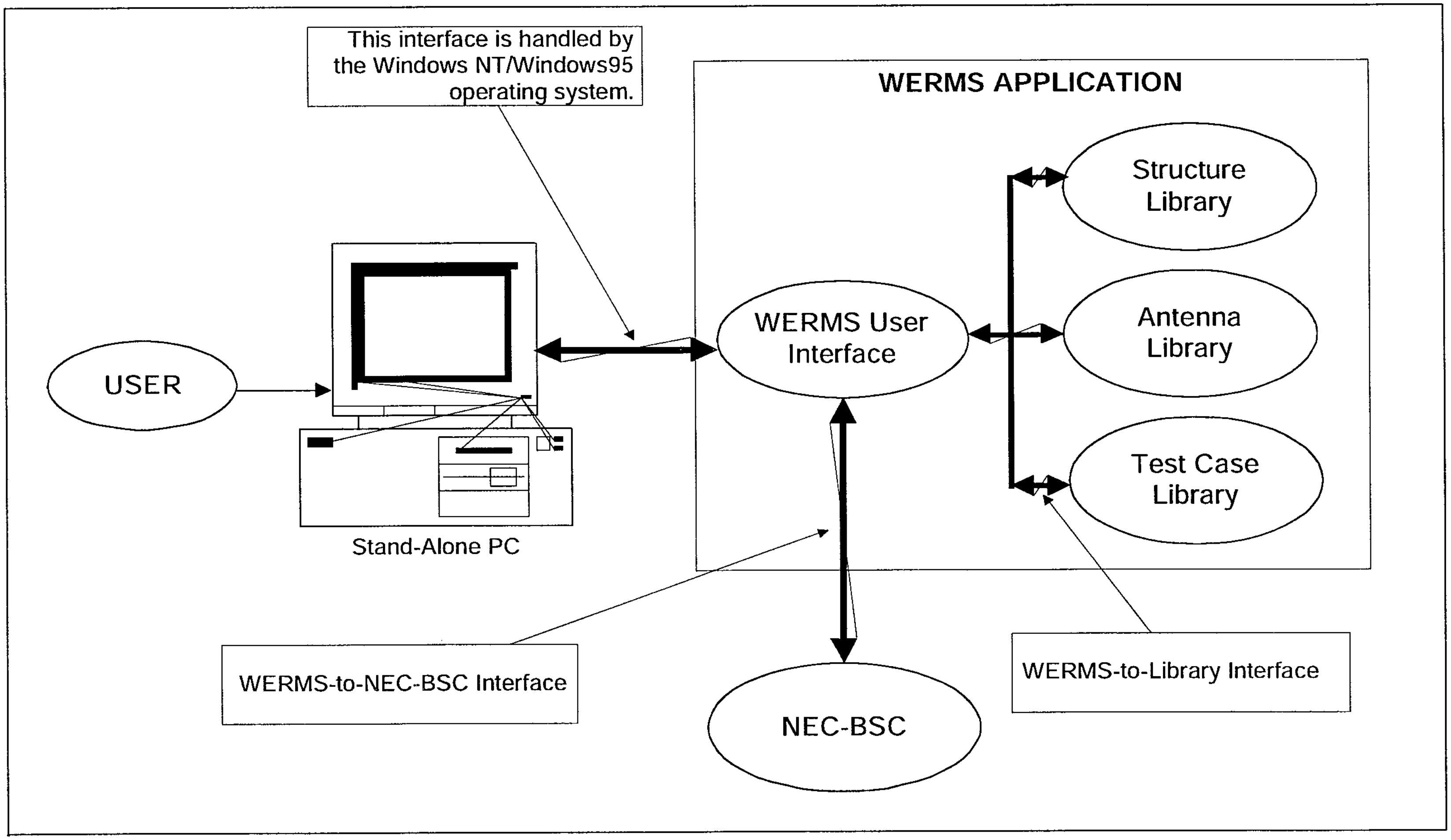
BRIEFING OVERVIEW

- WERMS PROGRAM
- STRUCTURES
- ANTENNAS
- TEST CASES
- EXECUTE
- OUTPUT

PROGRAM OBJECTIVE

- Design and Develop an Accurate Computer Software Model to Simulate and Predict the RF Environment and Characterization of Anechoic Chambers
- Develop a Baseline Model of the PRIMES Chamber and Ancillary Equipment
- Develop Baseline Models of the Systems and Antennas Tested at the PRIMES Facility
- Develop a User-Friendly PC Workstation to Configure Test Scenarios and Perform Simulated Tests Runs

WERMS FUNCTIONAL DIAGRAM



This interface is handled by the Windows NT/Windows95 operating system.
WERMS APPLICATION
USER
Stand-Alone PC
WERMS User Interface
Structure Library
Antenna Library
Test Case Library
WERMS-to-NEC-BSC Interface
NEC-BSC
WERMS-to-Library Interface

SOFTWARE SUPPORT ENVIRONMENT(SSE) DESIGN OBJECTIVES

- User Interactions Revolve Around Graphical View of Structure(s) Selected for Test Case
- Provide for Efficient Placement/Orientation of Simulation Objects in Test Scenario
- Graphical Interface to Specify Execution Options
    - Output/Cut Selection, Field Terms, Etc.
- Easily Change Configuration/Execution Options
- Overlay Output for Comparisons
    - Polar, Linear, Volumetric, Ray Tracing Plots

WERMS SSE DIAGRAM

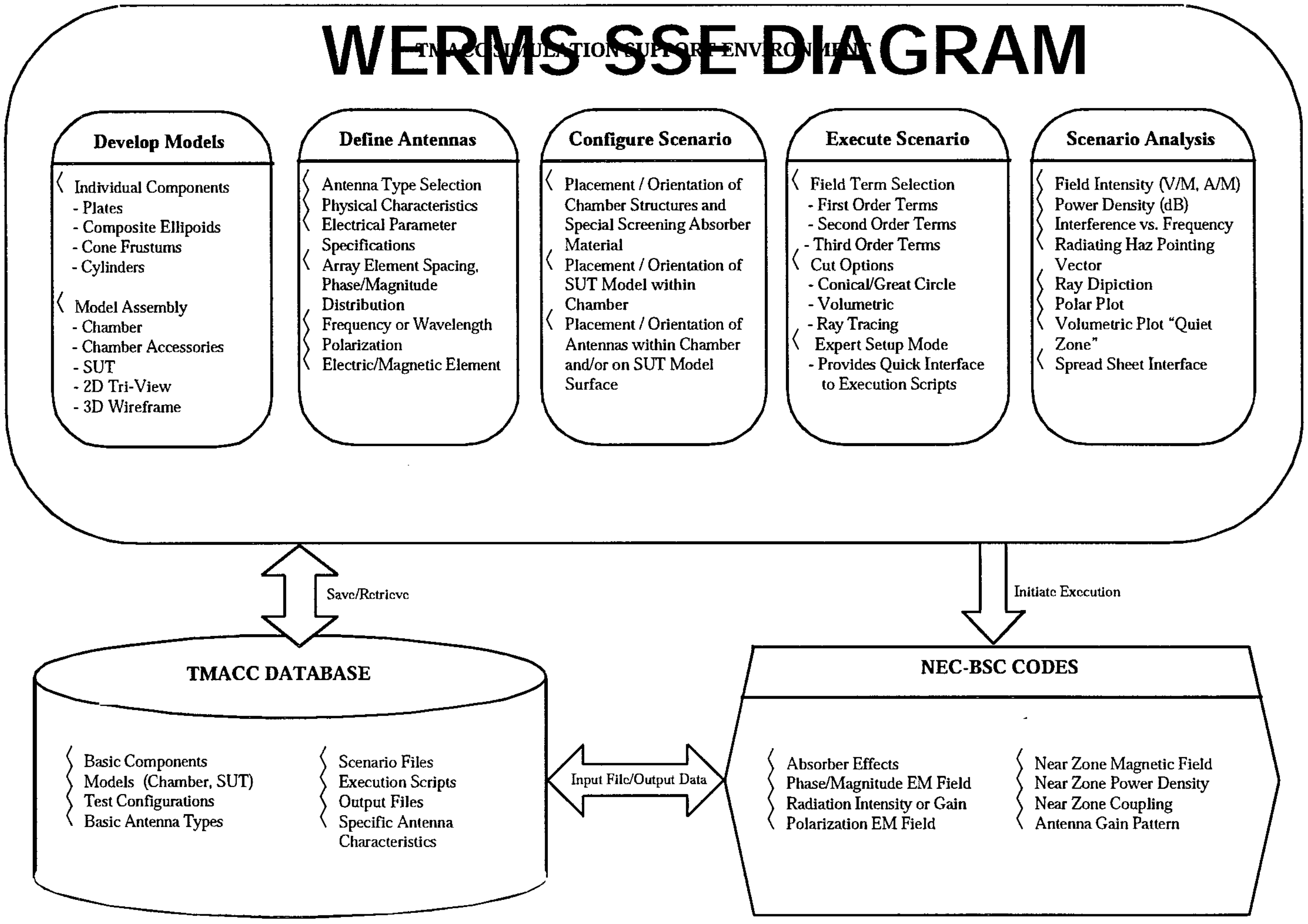
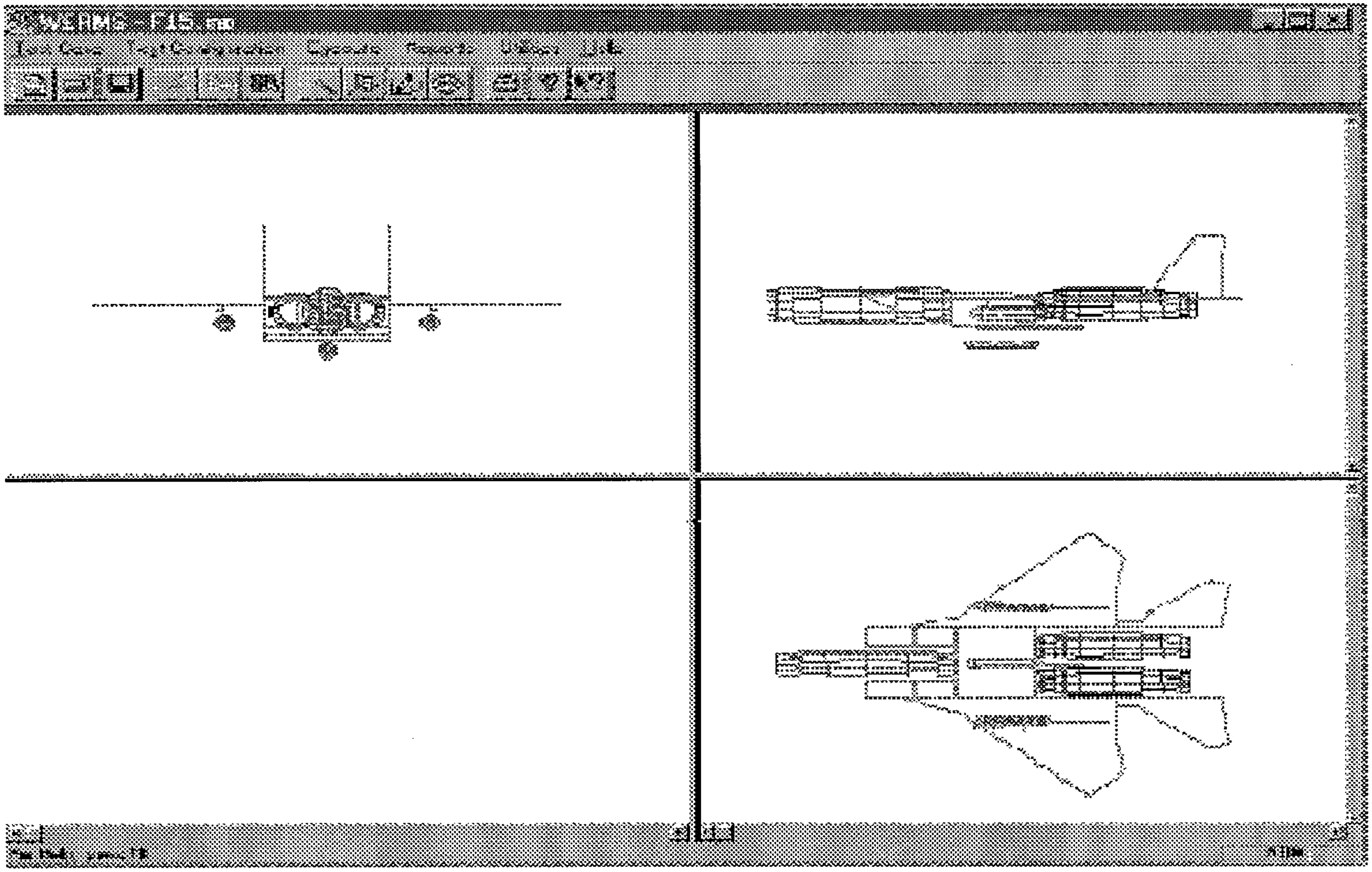


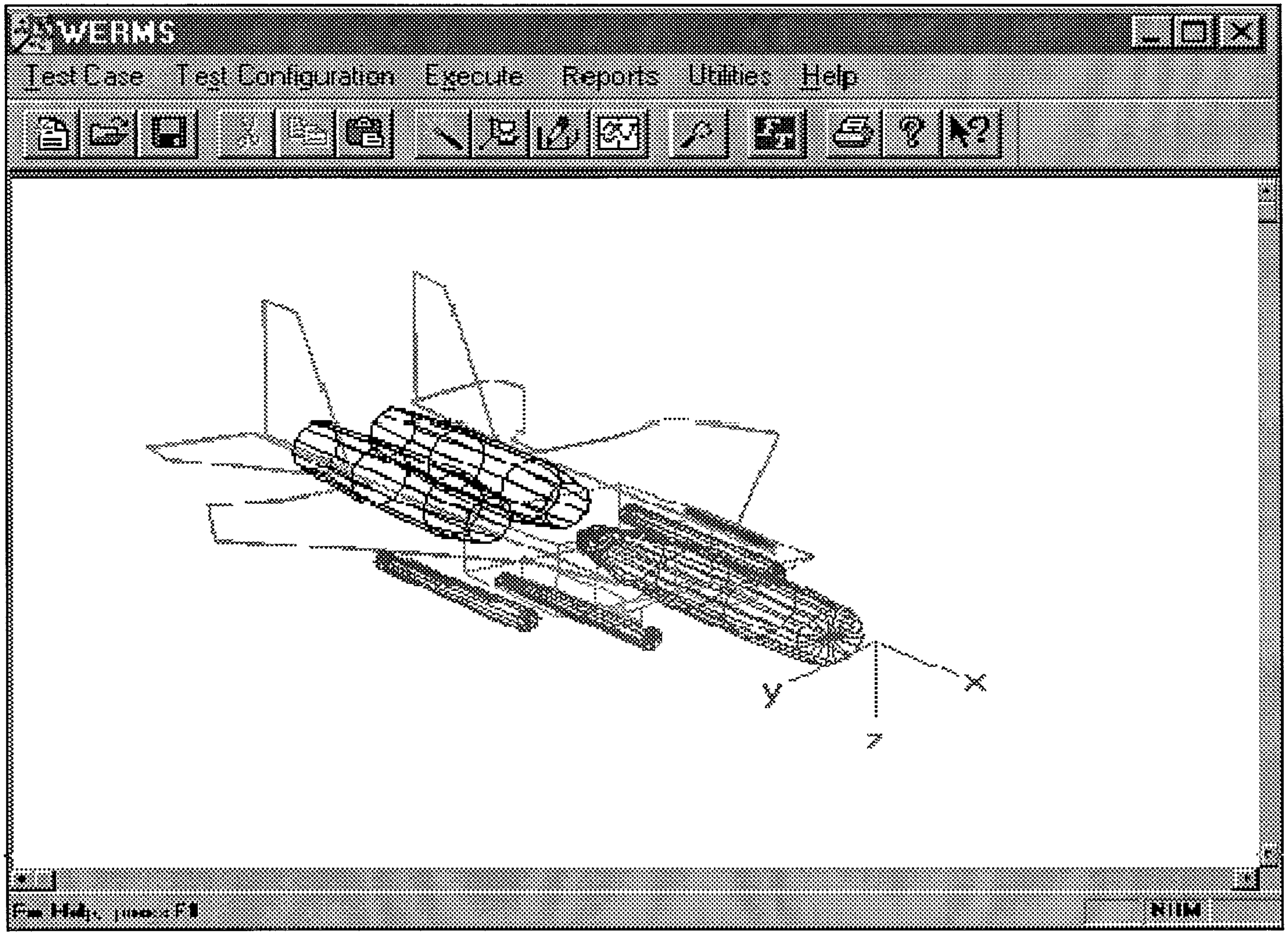
Figure 4 - TMACC Simulation Support Environment Block Diagram

QUAD VIEW DISPLAY

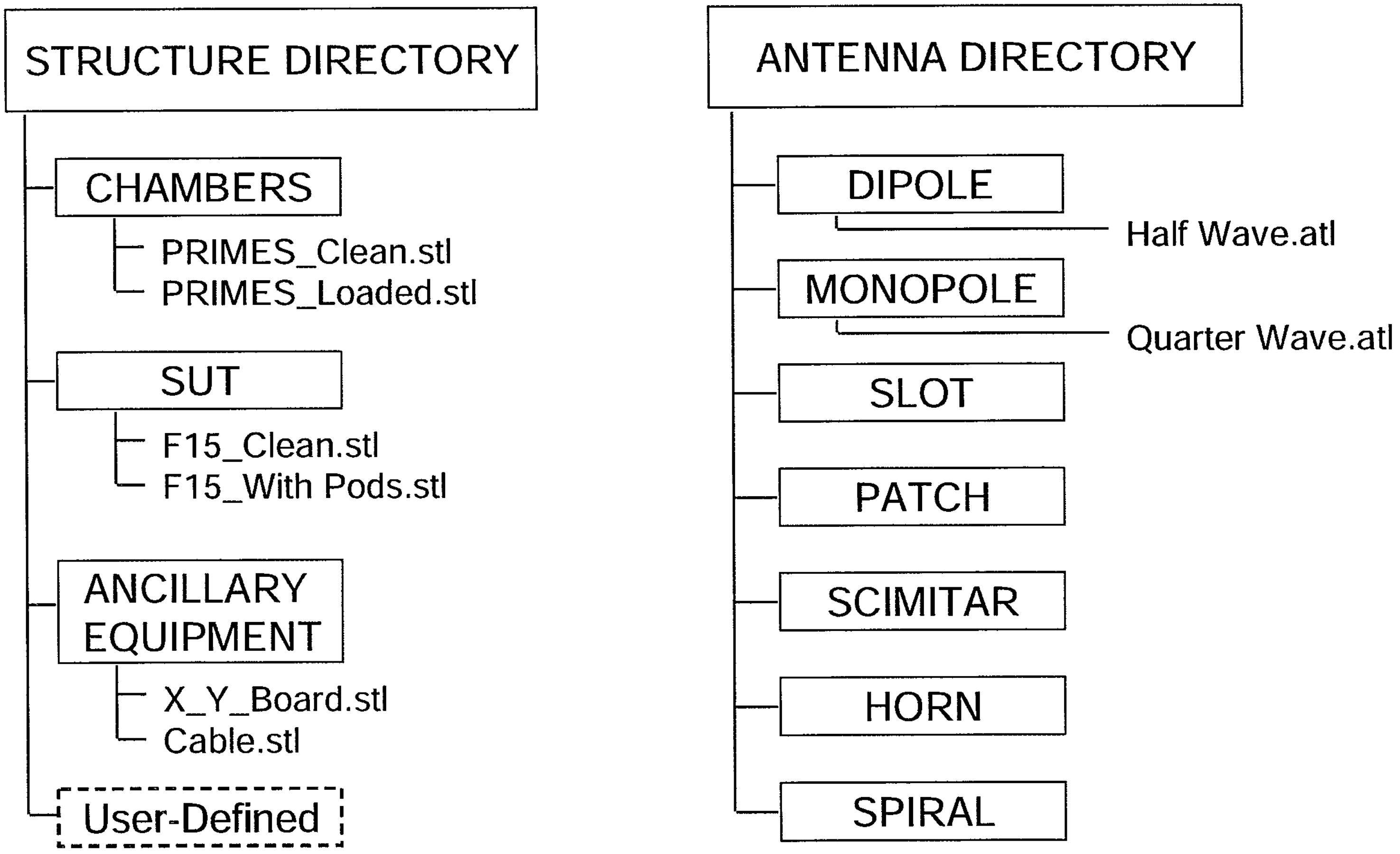


WERMS

WIREFRAME DISPLAY



STRUCTURE AND ANTENNA LIBRARY



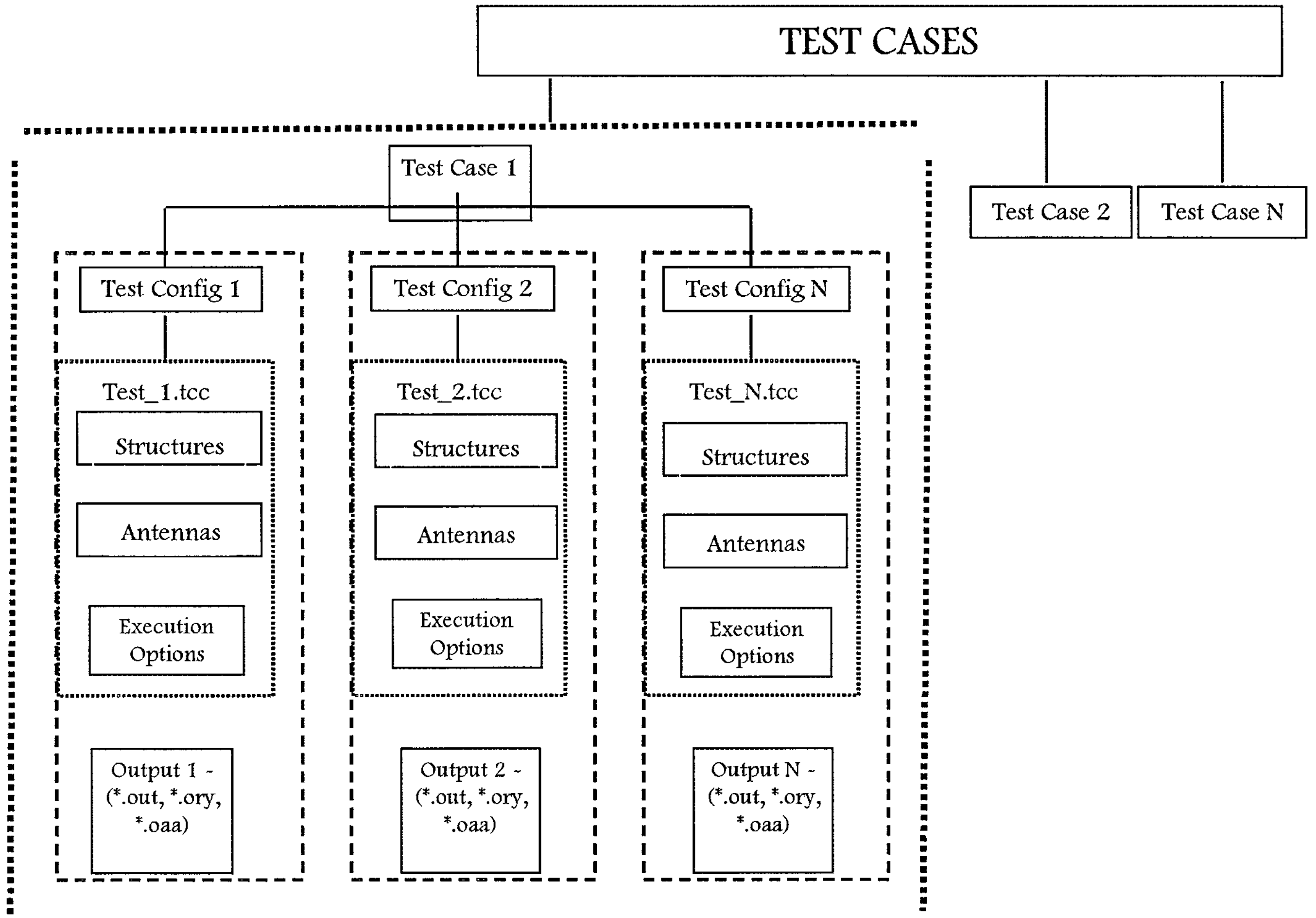
STRUCTURE DIRECTORY

- CHAMBERS
  - PRIMES_Clean.stl
  - PRIMES_Loaded.stl
- SUT
  - F15_Clean.stl
  - F15_With Pods.stl
- ANCILLARY EQUIPMENT
  - X_Y_Board.stl
  - Cable.stl
- User-Defined

ANTENNA DIRECTORY

- DIPOLE
  - Half Wave.atl
- MONOPOLE
  - Quarter Wave.atl
- SLOT
- PATCH
- SCIMITAR
- HORN
- SPIRAL

TEST CASE LIBRARY



TEST CASES

Test Case 1

Test Case 2

Test Case N

Test Config 1

Test_1.tcc

Structures

Antennas

Execution Options

Output 1 - (*.out, *.ory, *.oaa)

Test Config 2

Test_2.tcc

Structures

Antennas

Execution Options

Output 2 - (*.out, *.ory, *.oaa)

Test Config N

Test_N.tcc

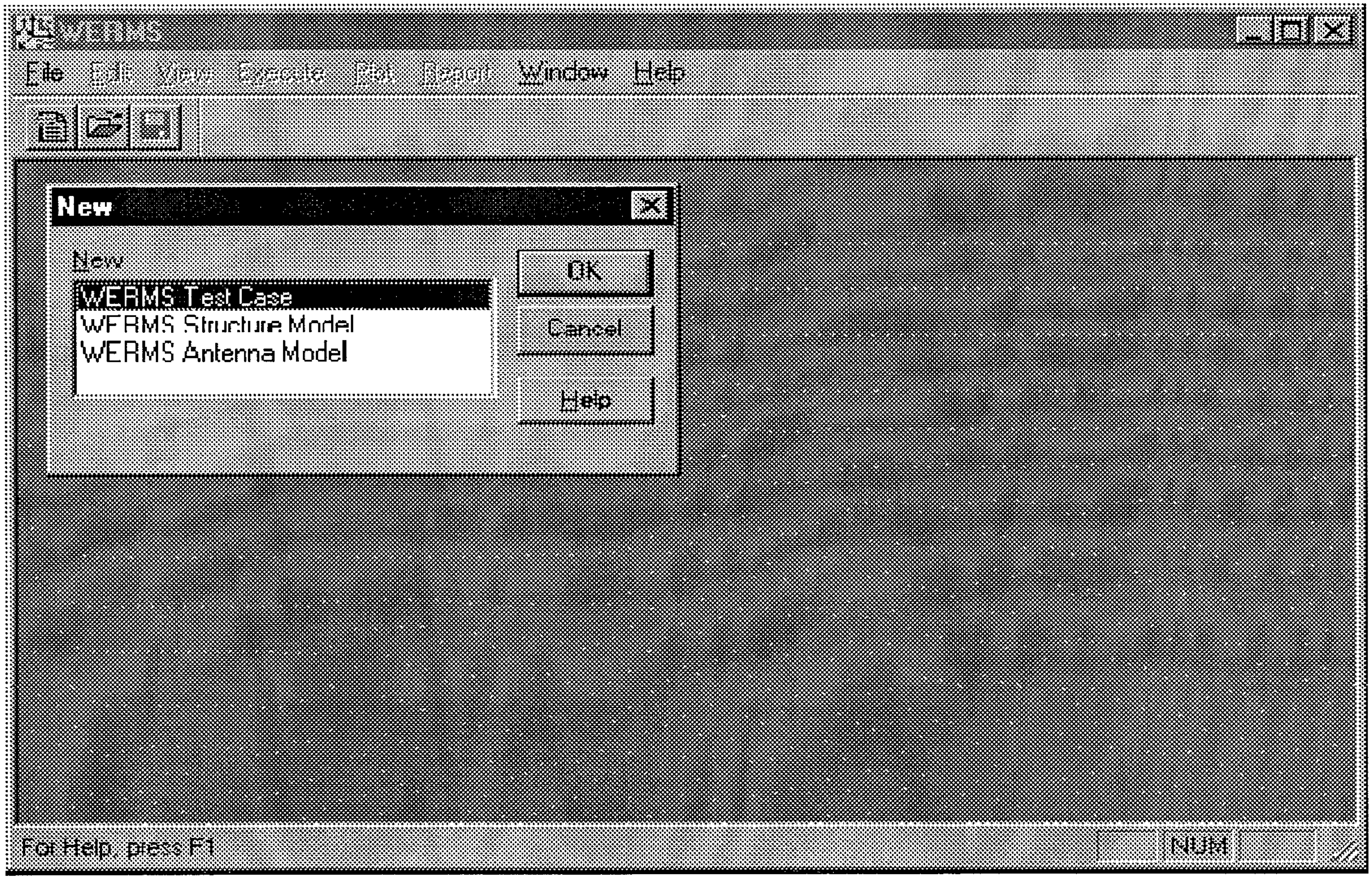
Structures

Antennas

Execution Options

Output N - (*.out, *.ory, *.oaa)

MODE SELECTION

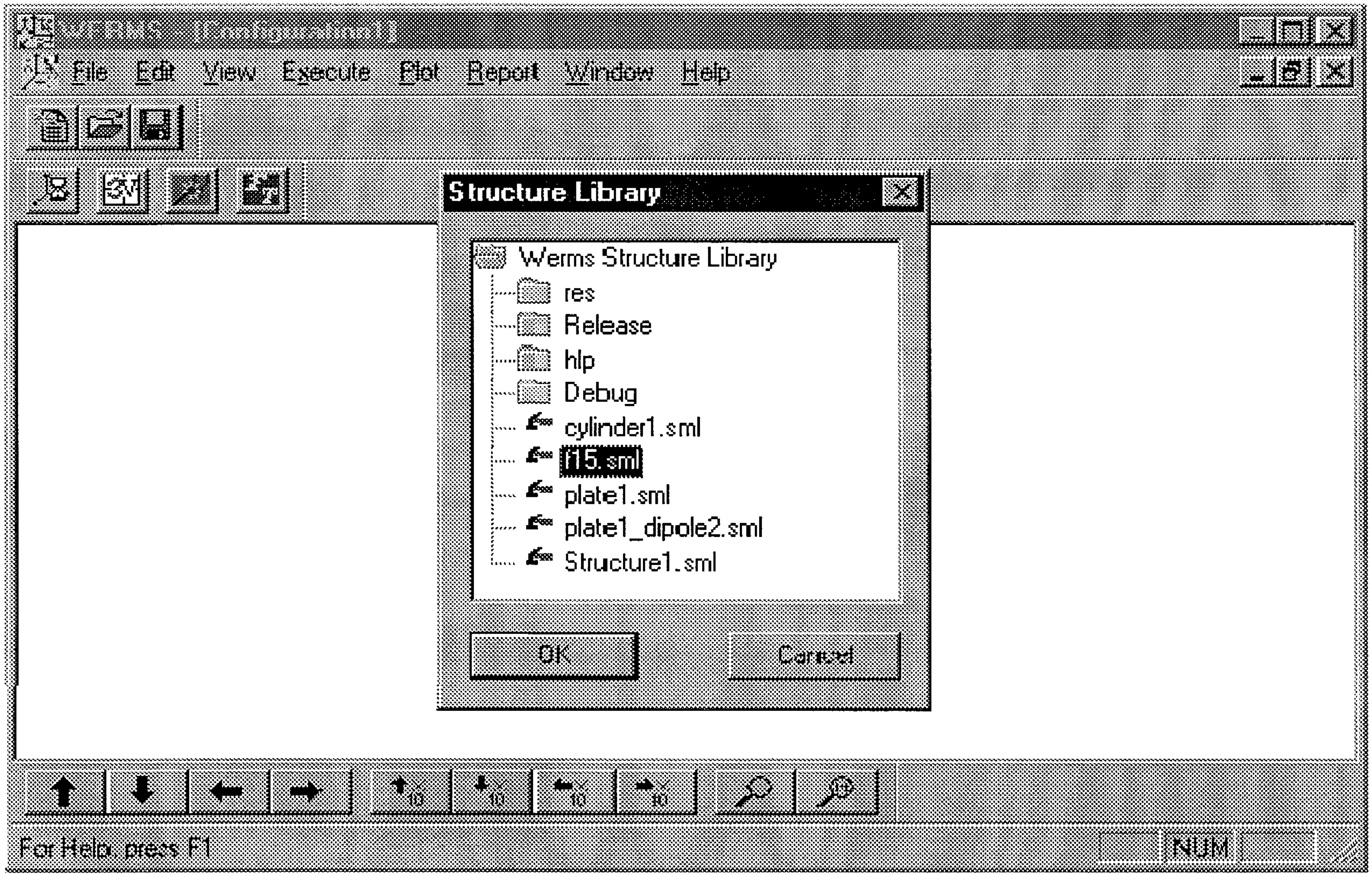


New

- WERMS Test Case
- WERMS Structure Model
- WERMS Antenna Model

WERMS

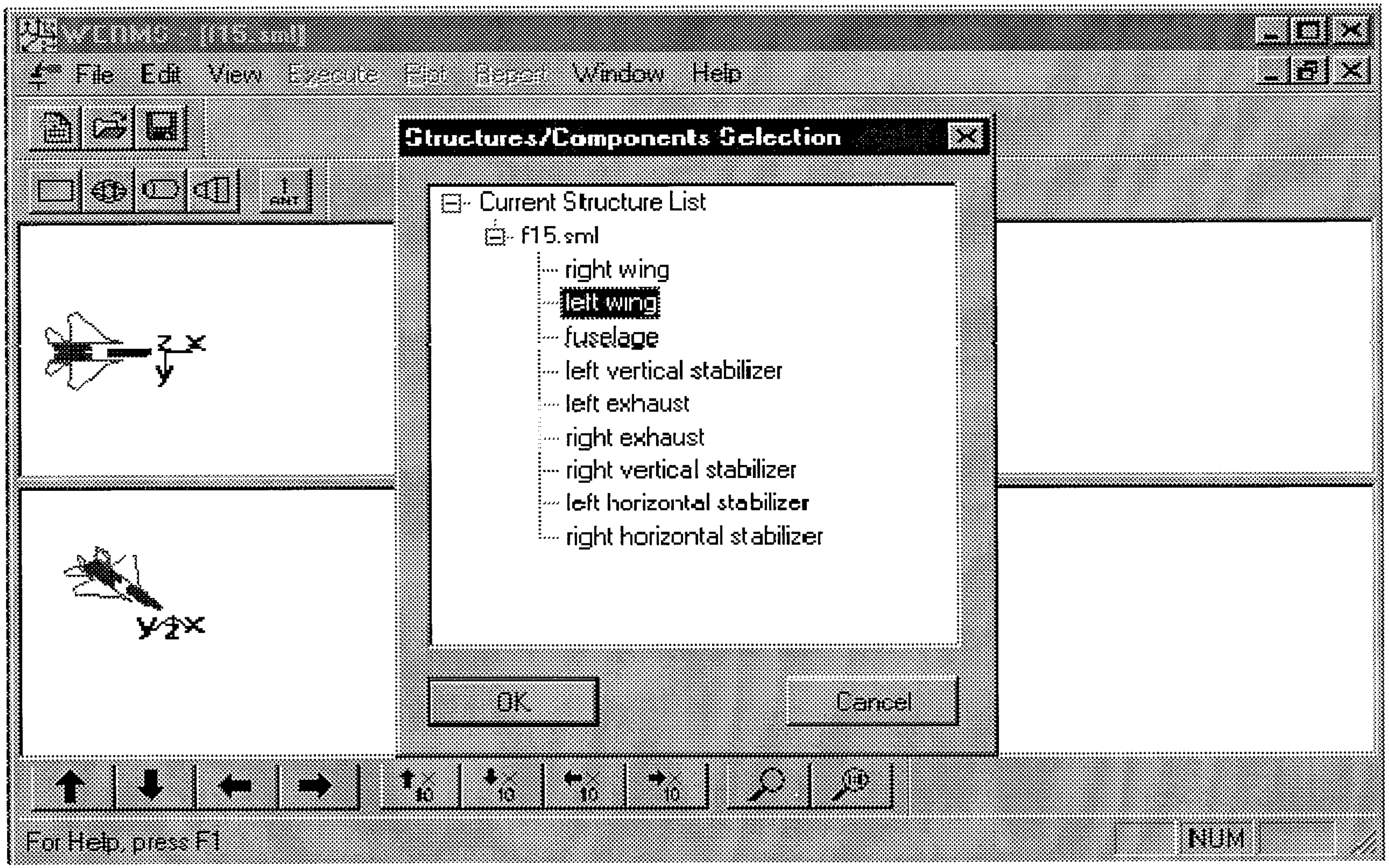
STRUCTURE LIBRARY SELECTION



Werms Structure Library
- res
- Release
- hlp
- Debug
- cylinder1.sml
- f15.sml
- plate1.sml
- plate1_dipole2.sml
- Structure1.sml

WERMS

STRUCTURE COMPONENT SELECTION



Structures/Components Selection

- Current Structure List
  - f15.sml
    - right wing
    - left wing
    - fuselage
    - left vertical stabilizer
    - left exhaust
    - right exhaust
    - right vertical stabilizer
    - left horizontal stabilizer
    - right horizontal stabilizer

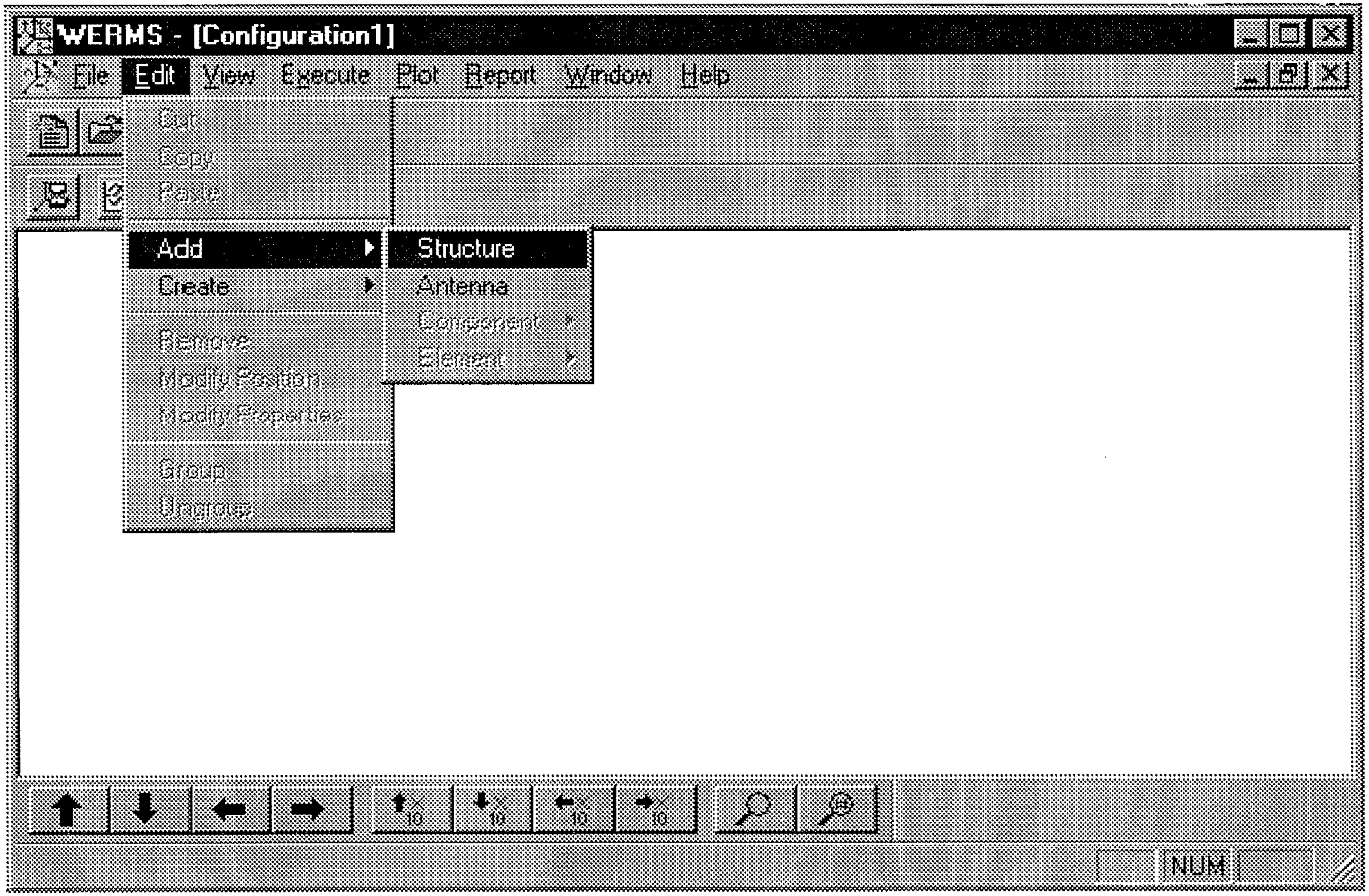
OK Cancel

WERMS

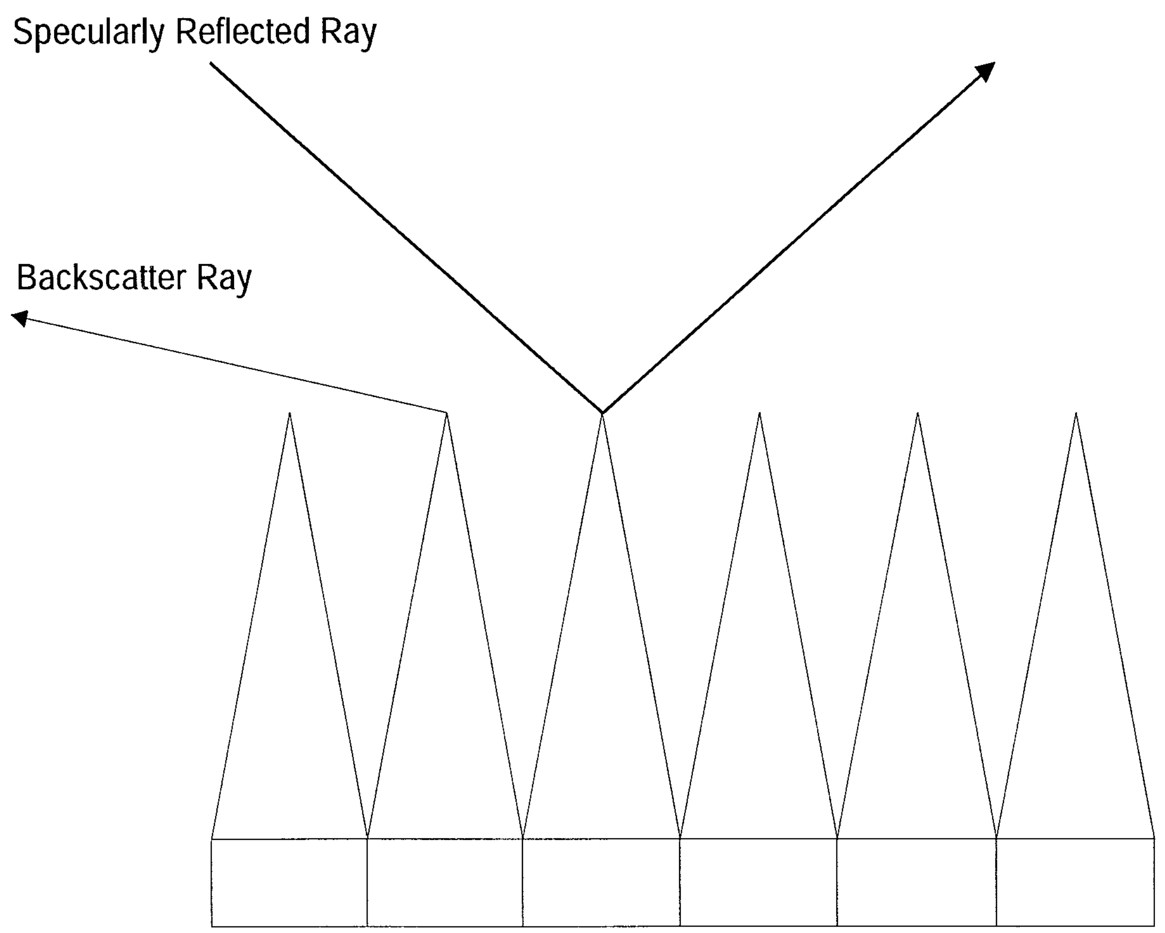
STRUCTURE MODELING

- FLAT PLATES
- ELLIPTICAL CYLINDERS
- ELLIPSOIDS
- CONE FRUSTUMS
- DIELECTRIC MATERIAL
  - COATING OR SOLID

ADD STRUCTURE



ABSORBER RAY MECHANISM



ABSORBER MODELING

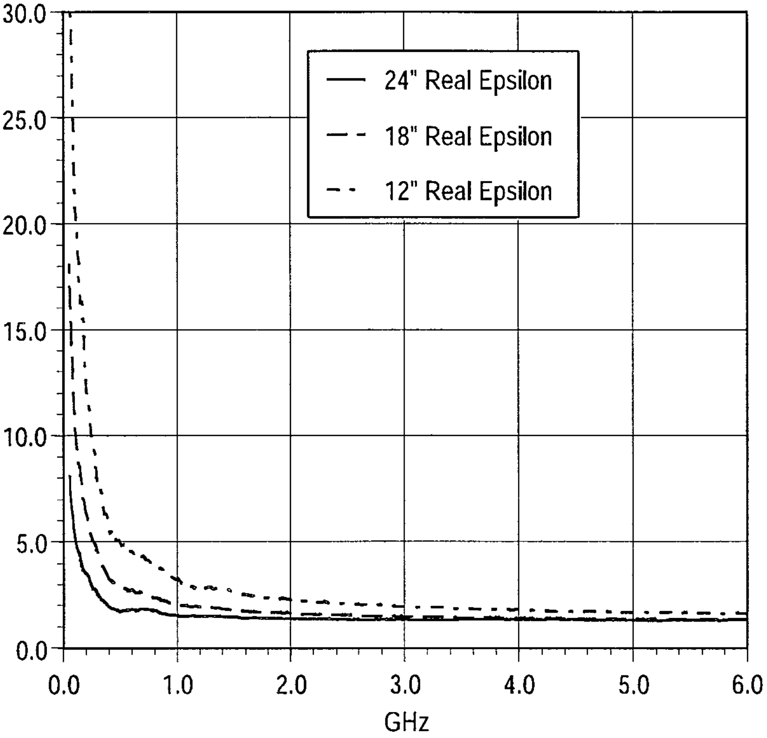
- Transmission Line Approximation (TLA) Algorithm used to represent absorber material by using an effective material concept to homogenize the material distribution in the periodic plane
- TLA was accurate at low frequency - wavelength long compared to period of structure
- TLA modified to make accurate at higher frequency by use of random point scattering technique and roughness factor

ABSORBER MEASUREMENTS

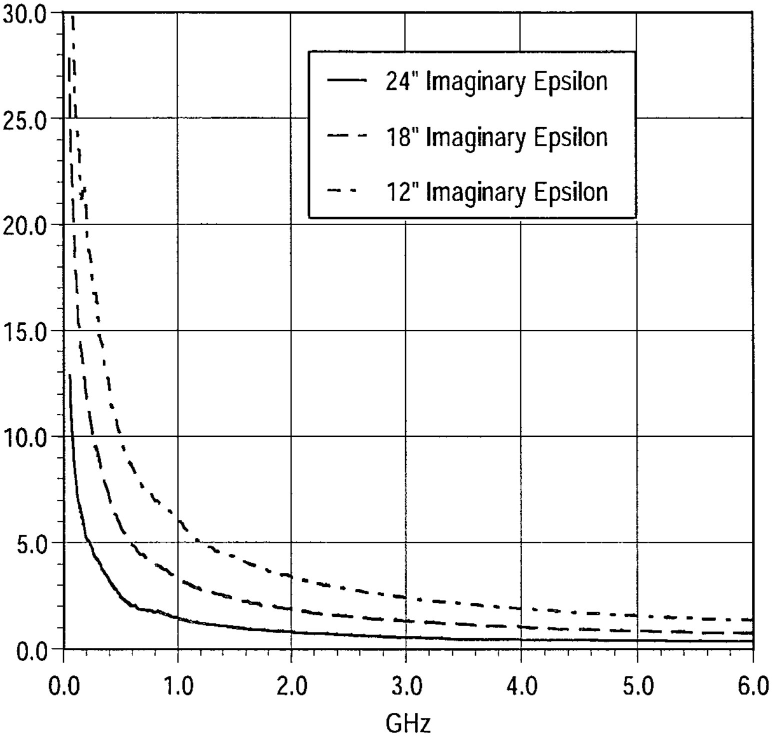
- Type
  - RANTAC - 5", 12", 18", 24" Pyramids
- Frequency
  - 2 GHz - 18 GHz in steps of 0.02 GHz
- Angle
  - + or - 45 Degrees of normal in steps of 0.2 degrees

ABSORBER MATERIAL PROPERTIES

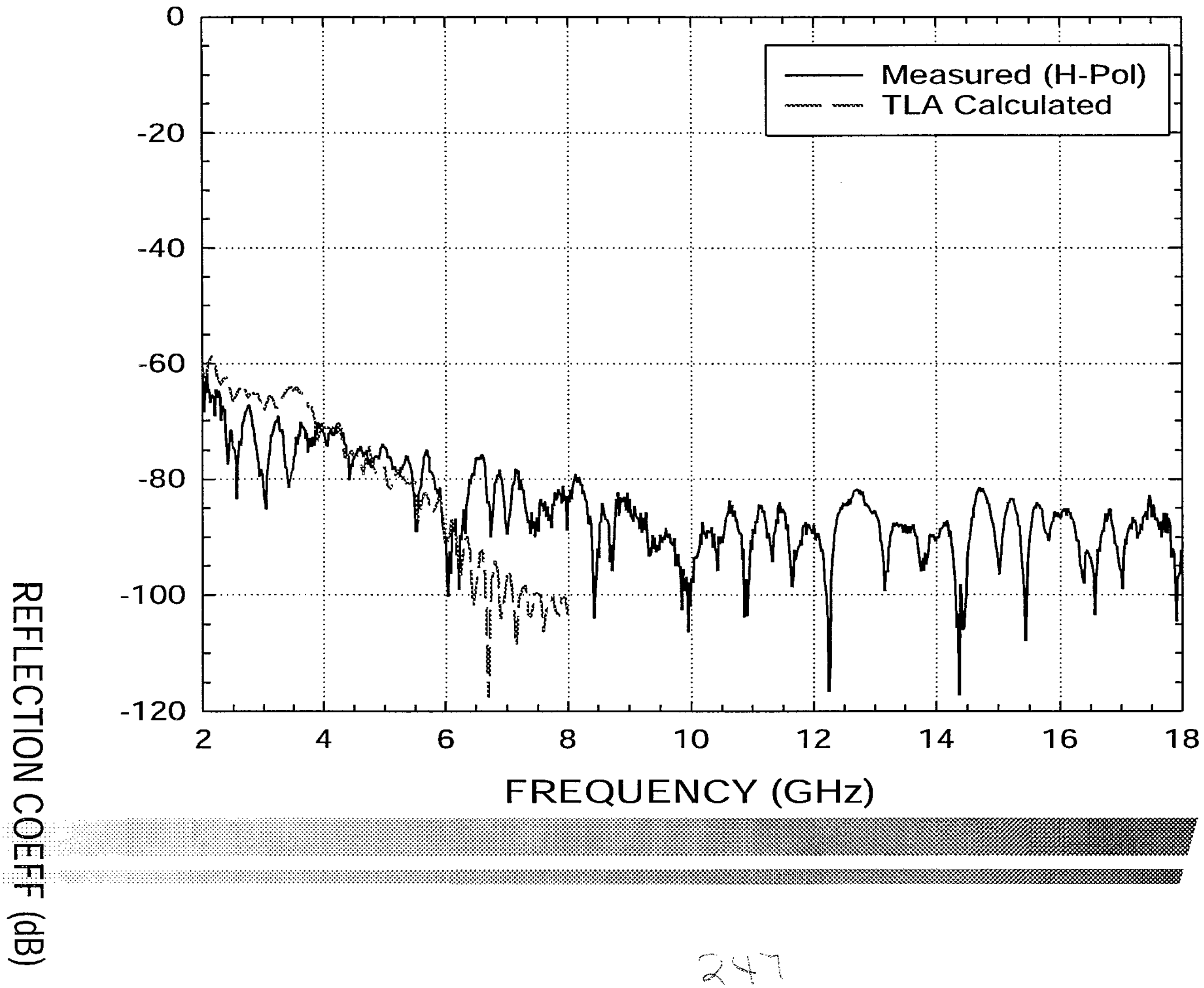
[redacted]



[redacted]

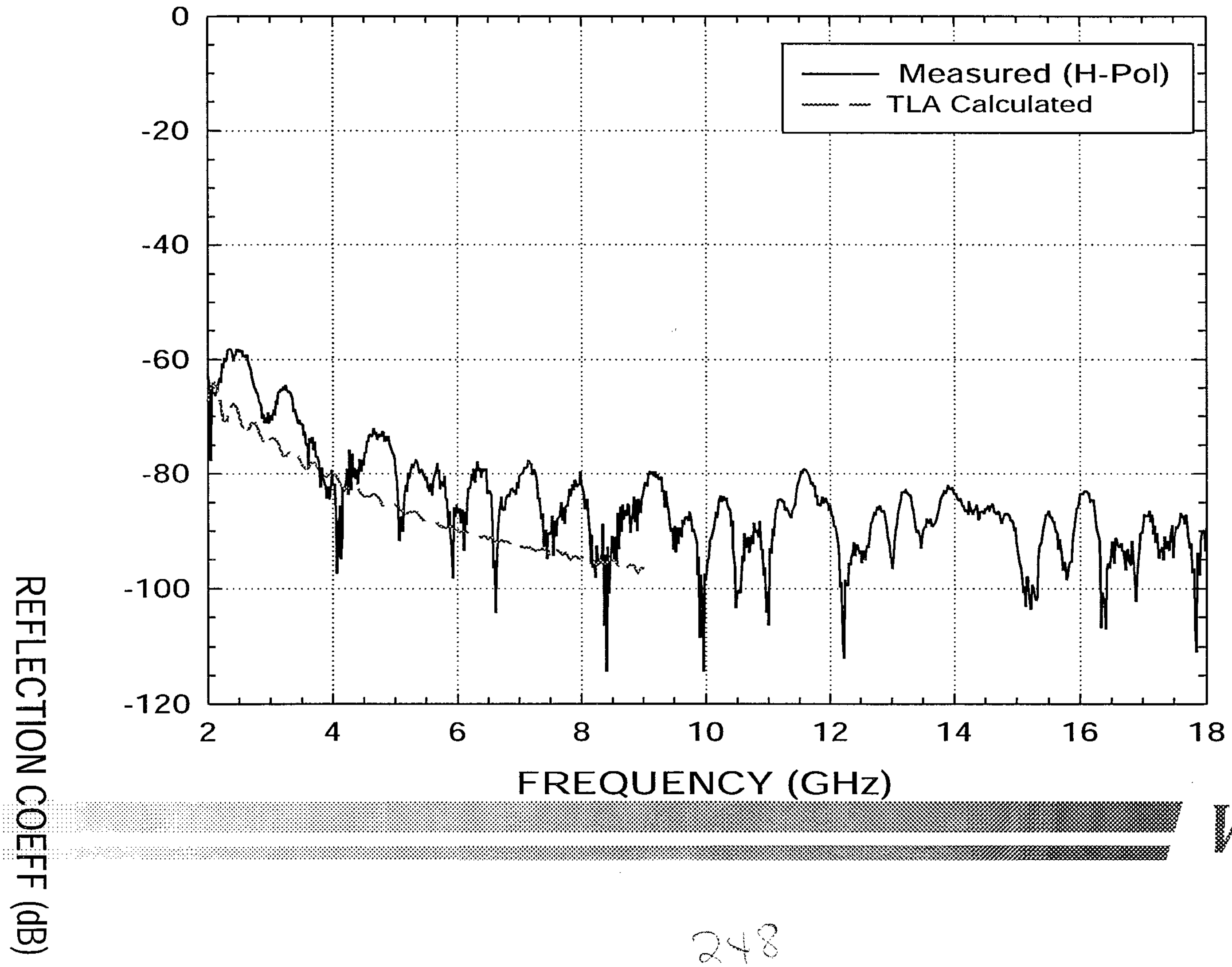


REFLECTION COEFFICIENT FOR 24-IN PYRAMID



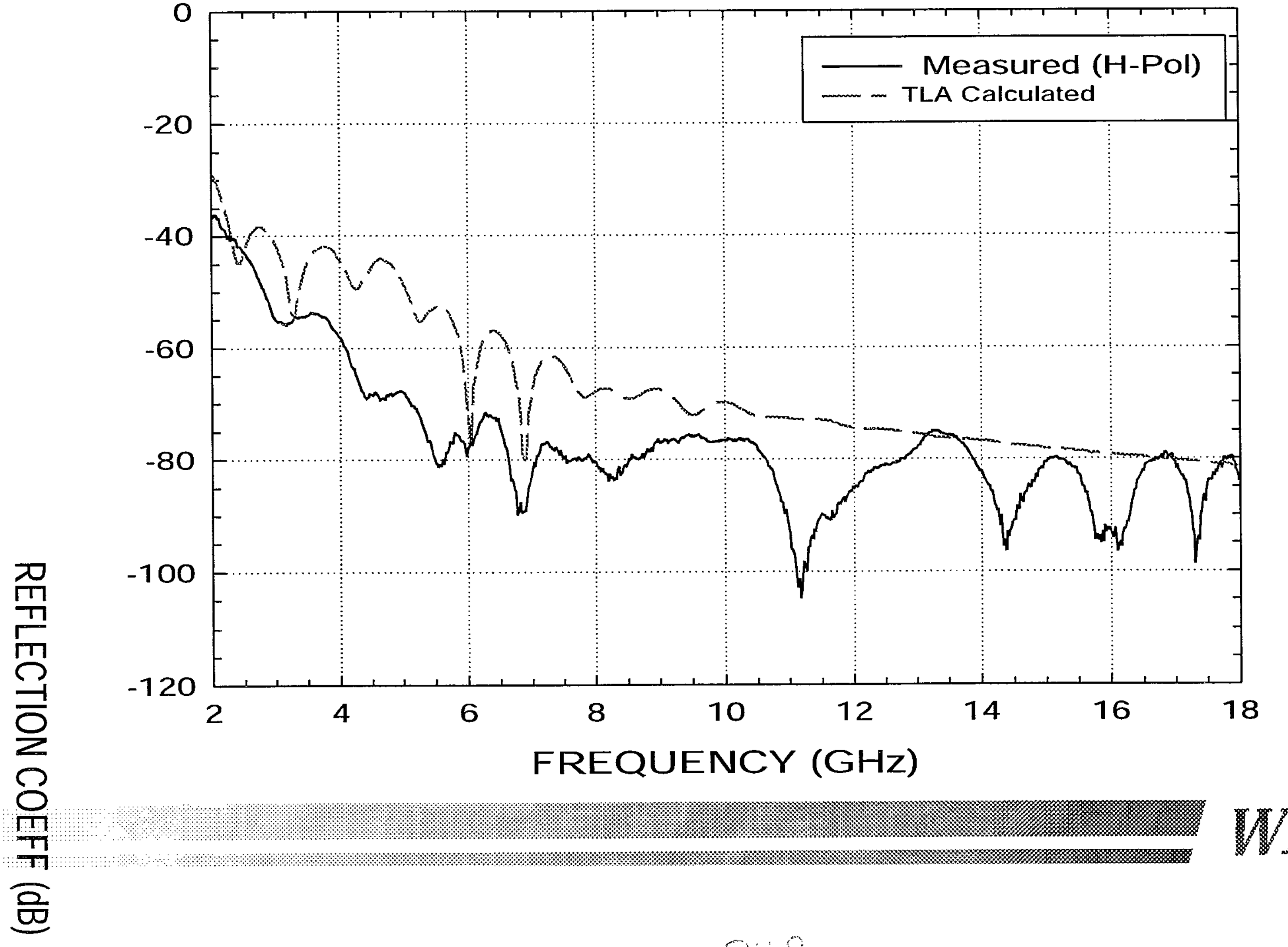
WERMS

REFLECTION COEFFICIENT FOR 18-IN PYRAMID



WERMS

REFLECTION COEFICIENT FOR 5-IN PYRAMID



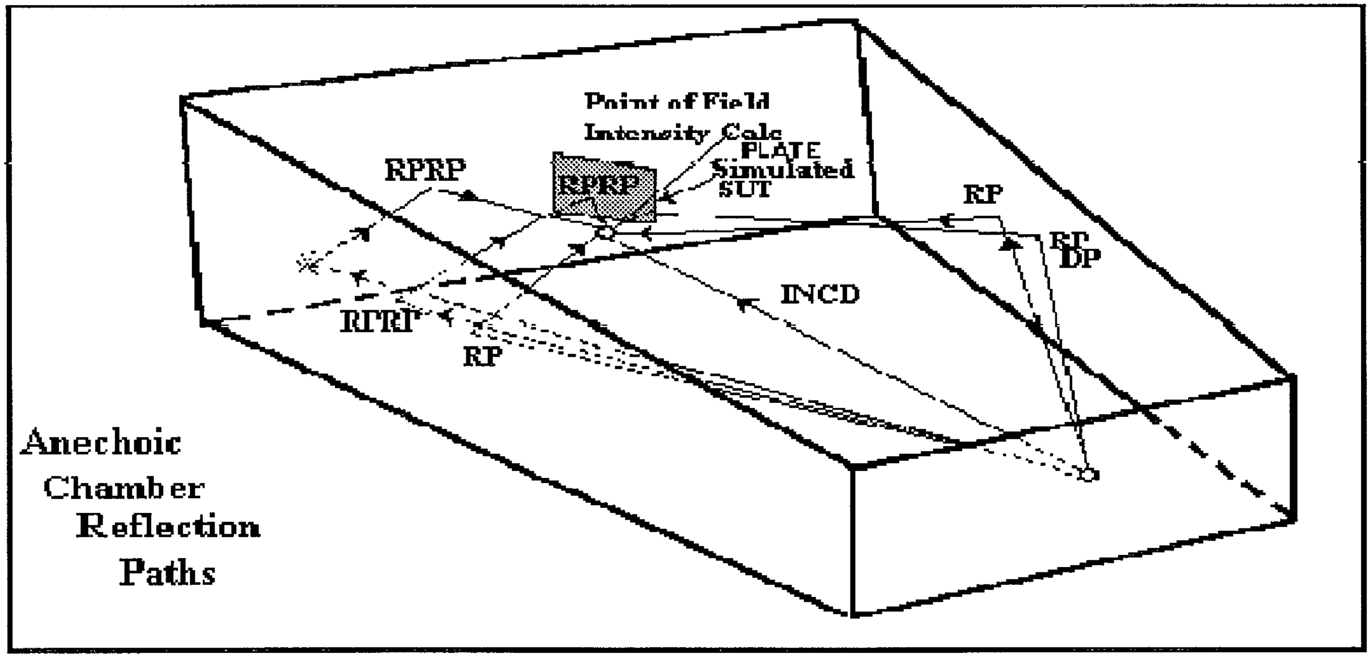
SYSTEM UNDER TEST (SUT) MODELING

- F-15
- F-16
- EW Pods
- Fuel Tanks
- Missiles
- Munitions Racks

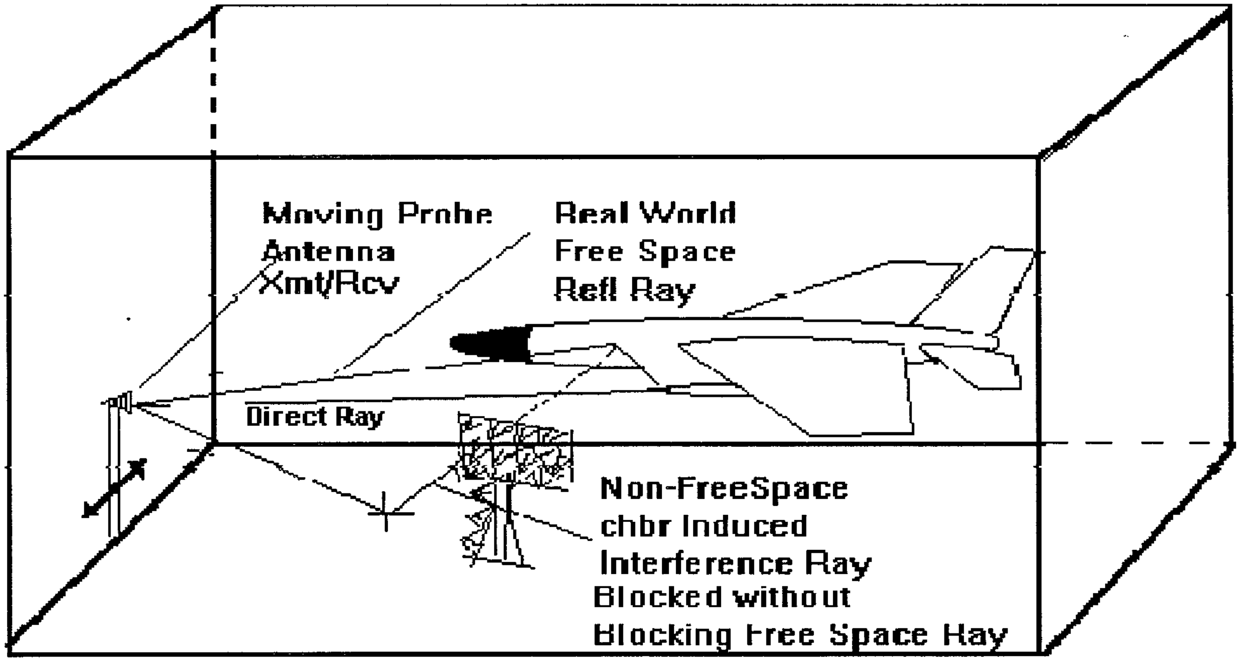
ANCILLARY EQUIPMENT MODELING

- F-15 Fire CONTROL RADAR SYSTEM
- RADIAL ARM TARGET TEST FIXTURE
- SUT HOST
- X-Y PLOTTING BOARD
- INTERFERENCE BLOCKING PLATES

PLATE DIFFRACTION/REFLECTION MULTIPLE FIELD TERMS

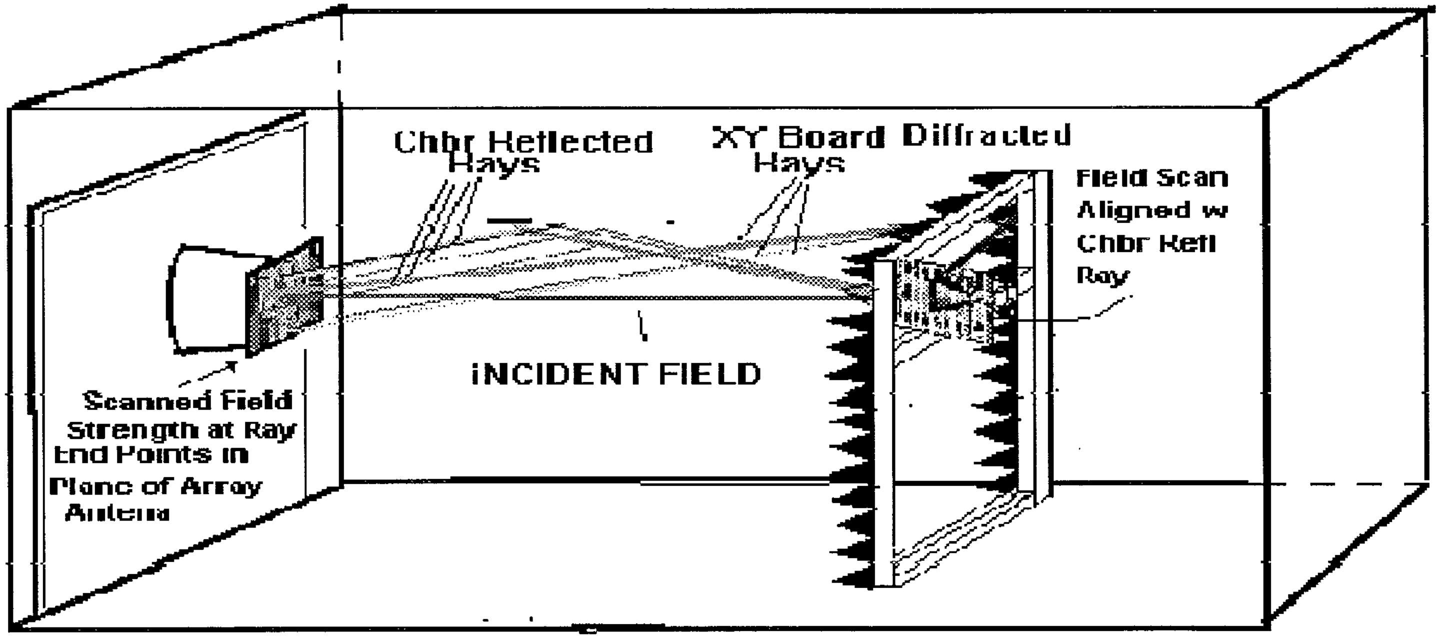


INTERFERENCE BLOCKING PLATES



WERMS AIDED CHAMBER INTERFERENCE BLOCKING

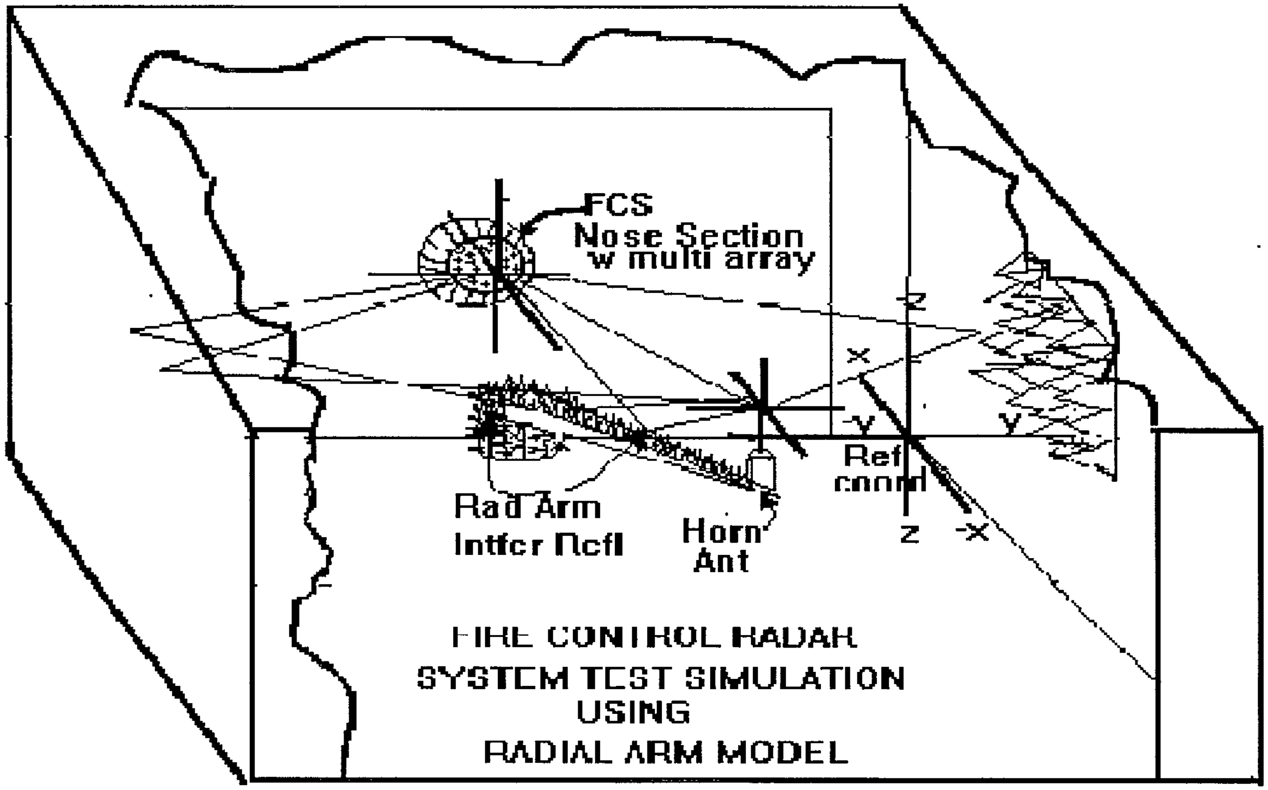
X-Y TARGET FIXTURE/CHAMBER INTERFERENCE SIMULATION


Chbr Reflected Rays
XY Board Diffracted Rays
Field Scan Aligned w Chbr Refl Ray
INCIDENT FIELD
Scanned Field Strength at Ray End Points in Plane of Array Antena

Plane of Linear Display Aligned with Chbr Induced Reflected Rays

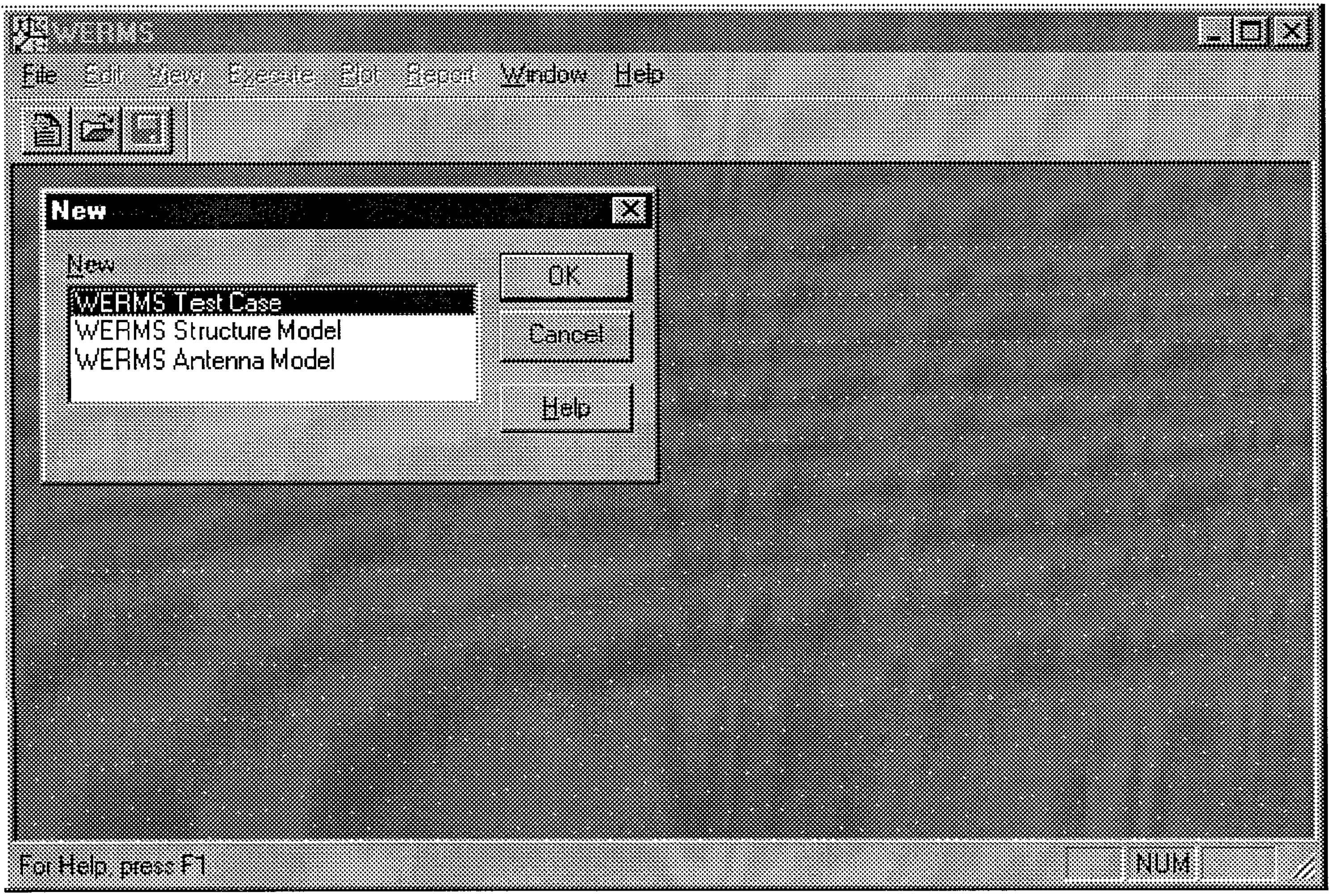
WERMS

CHAMBER/SUT/RADIAL ARM TARGET SCENARIO



FIRE CONTROL RADAR
SYSTEM TEST SIMULATION
USING
RADIAL ARM MODEL

ANTENNA MODELING



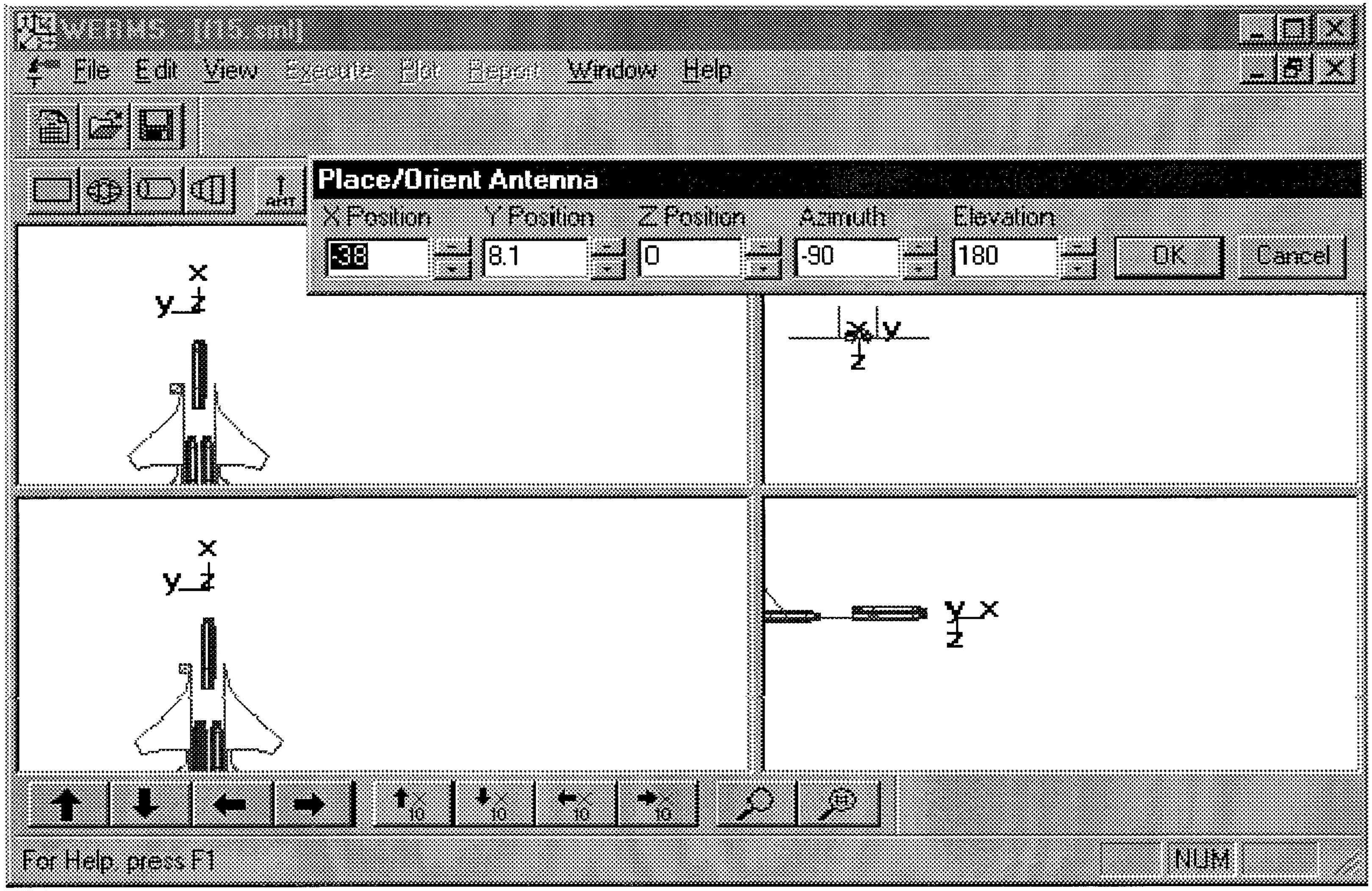
New

WERMS Test Case
WERMS Structure Model
WERMS Antenna Model

ANTENNA TYPES

- MONOPOLE
- DIPOLE
- SCIMITAR
- SLOT
- MICROPATCH
- HORN
- SPIRAL

ANTENNA PLACEMENT/ORIENTATION



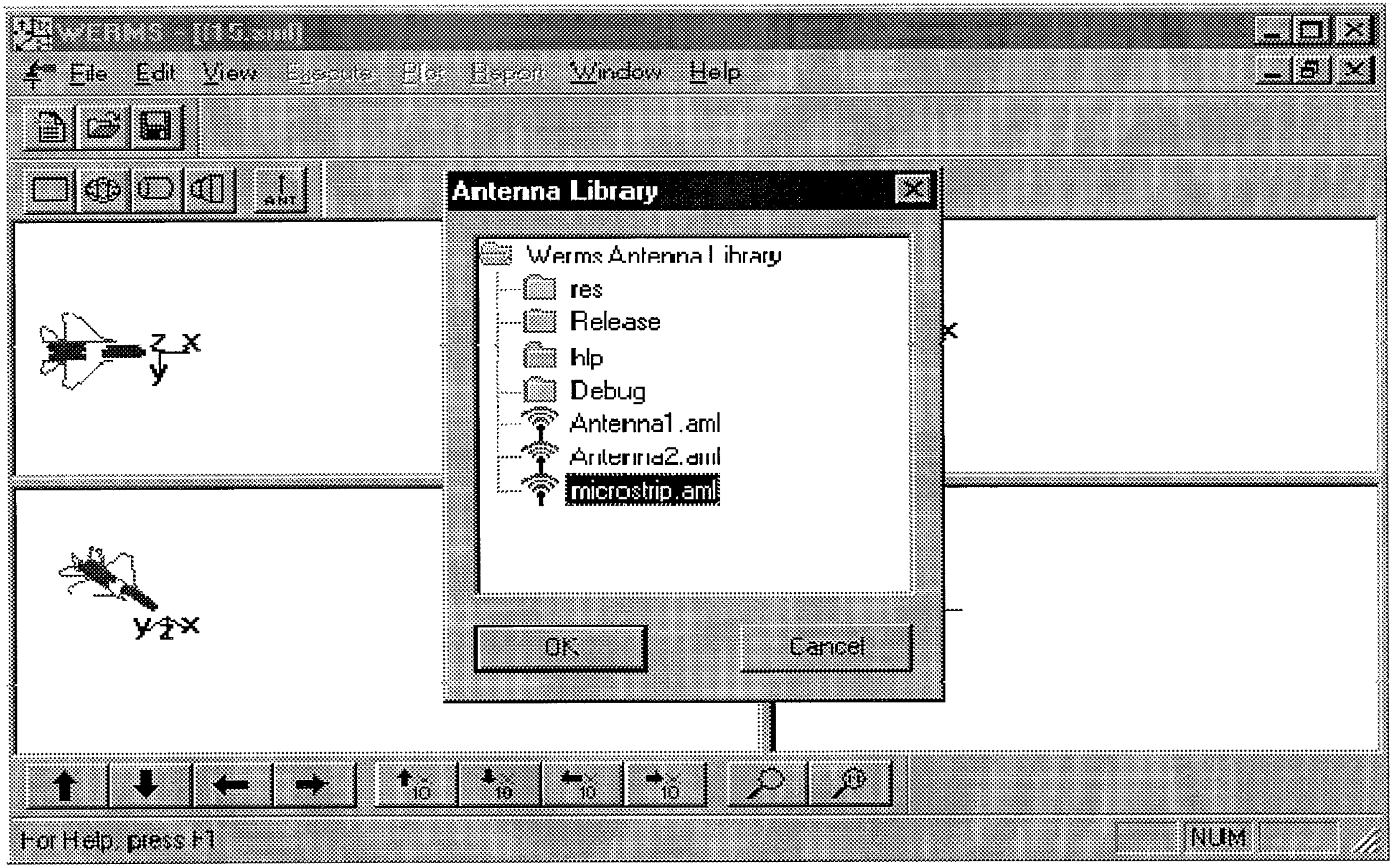
Place/Orient Antenna

X Position	Y Position	Z Position	Azimuth	Elevation
-38	8.1	0	-90	180

OK Cancel

WERMS

EDIT/ADD ANTENNA

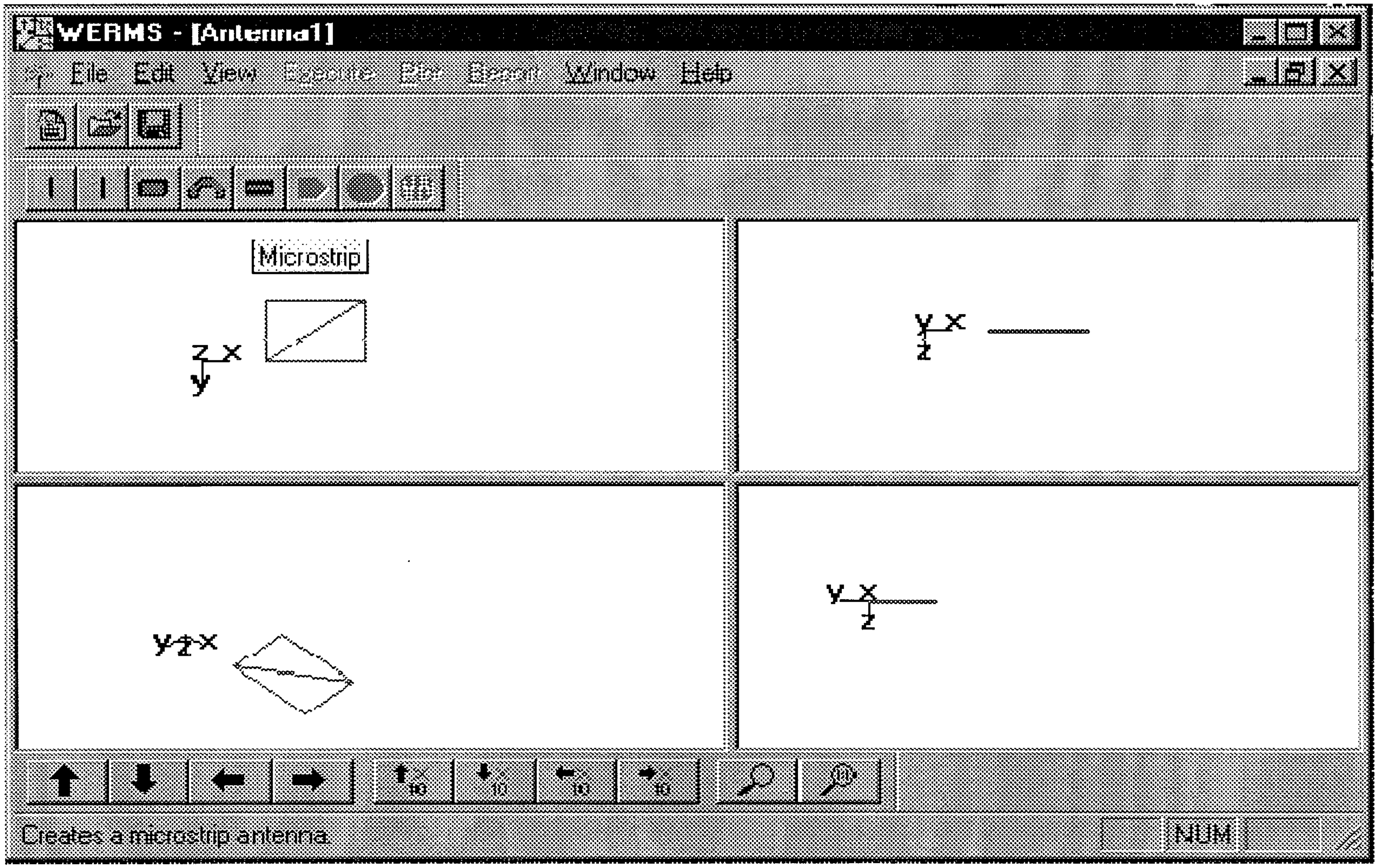


Antenna Library

- Werms Antenna Library
  - res
  - Release
  - hlp
  - Debug
  - Antenna1.aml
  - Antenna2.aml
  - microstrip.aml

WERMS

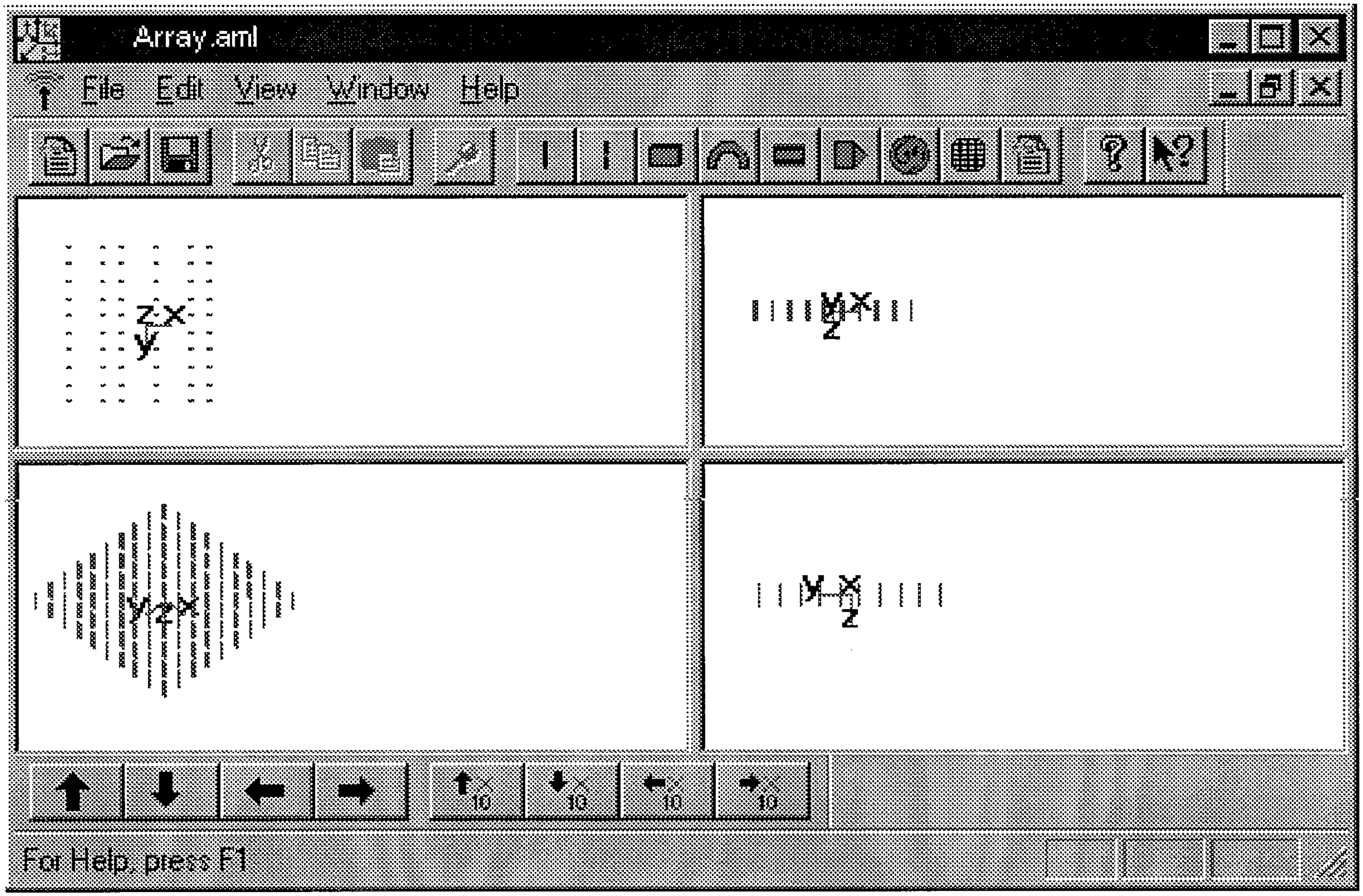
MICROSTRIP ANTENNA MODELING



Creates a microstrip antenna.

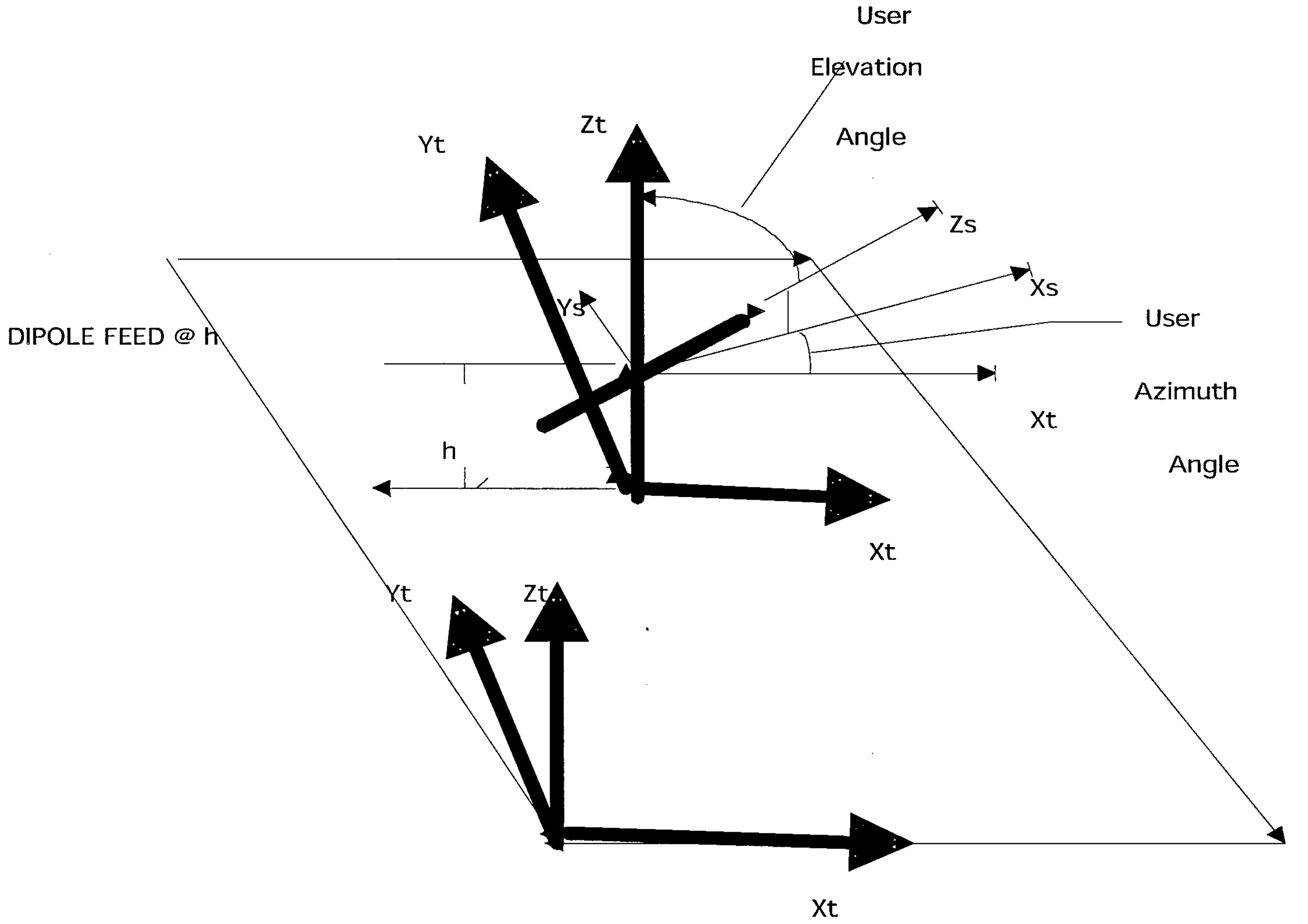
WERMS

ARRAY ANTENNA MODELING

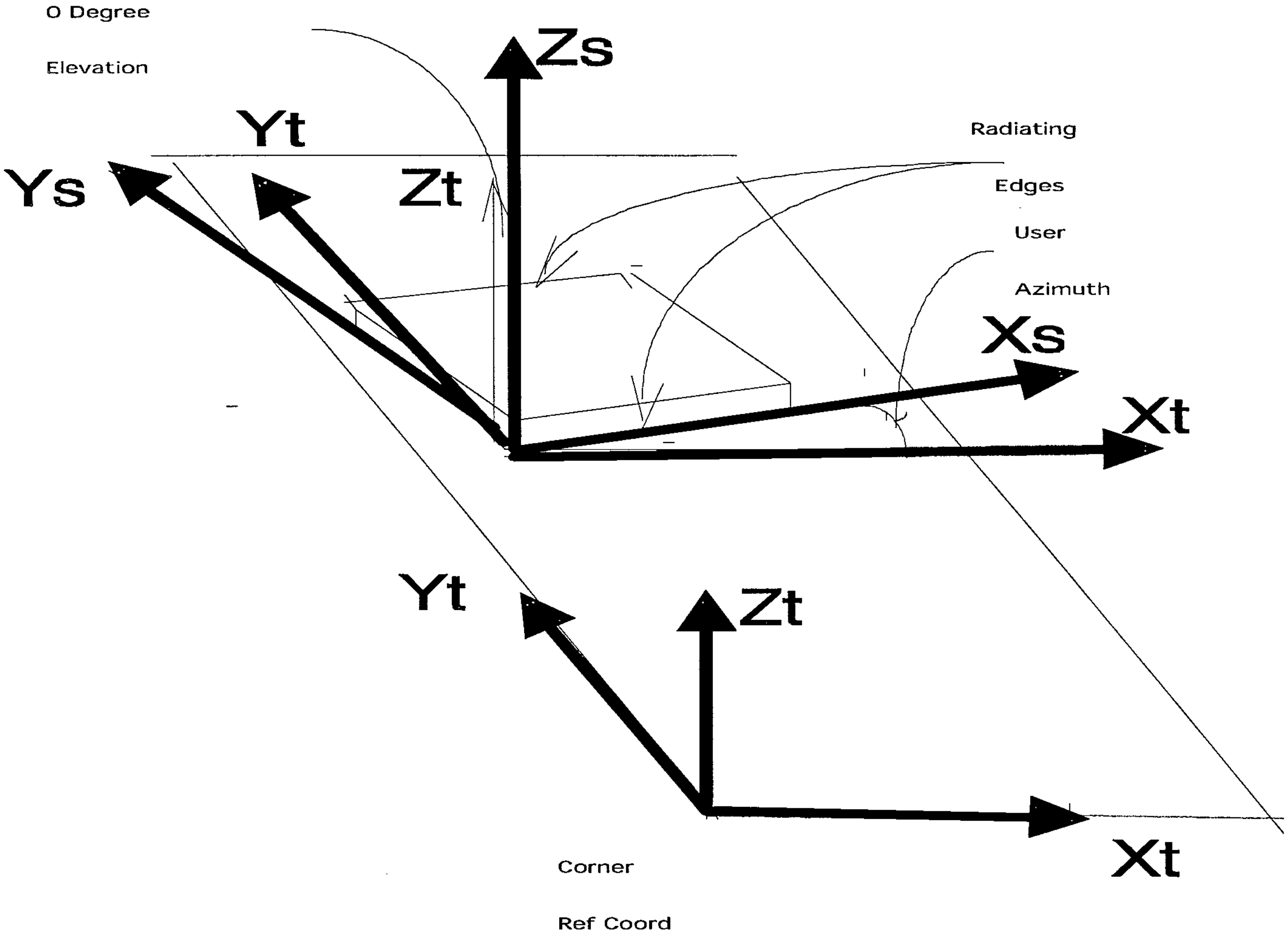


For Help, press F1

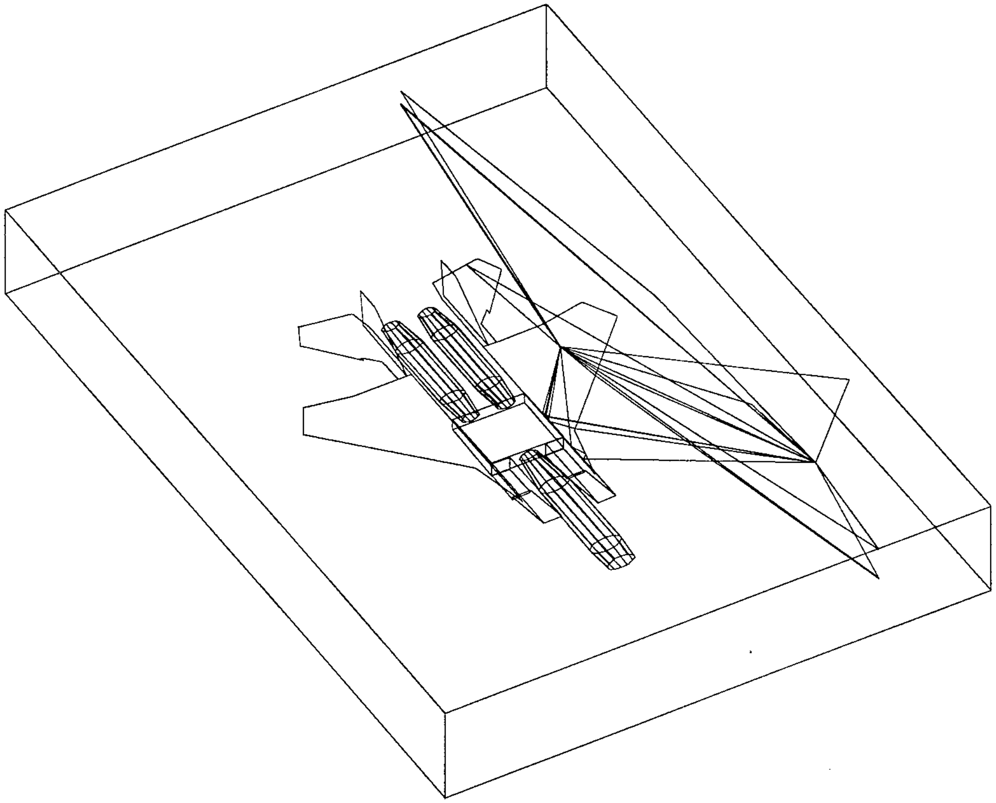
DIPOLE ORIENTATION ON FLAT PLATE



MICROPATCH ORIENTATION - FLAT PLATE

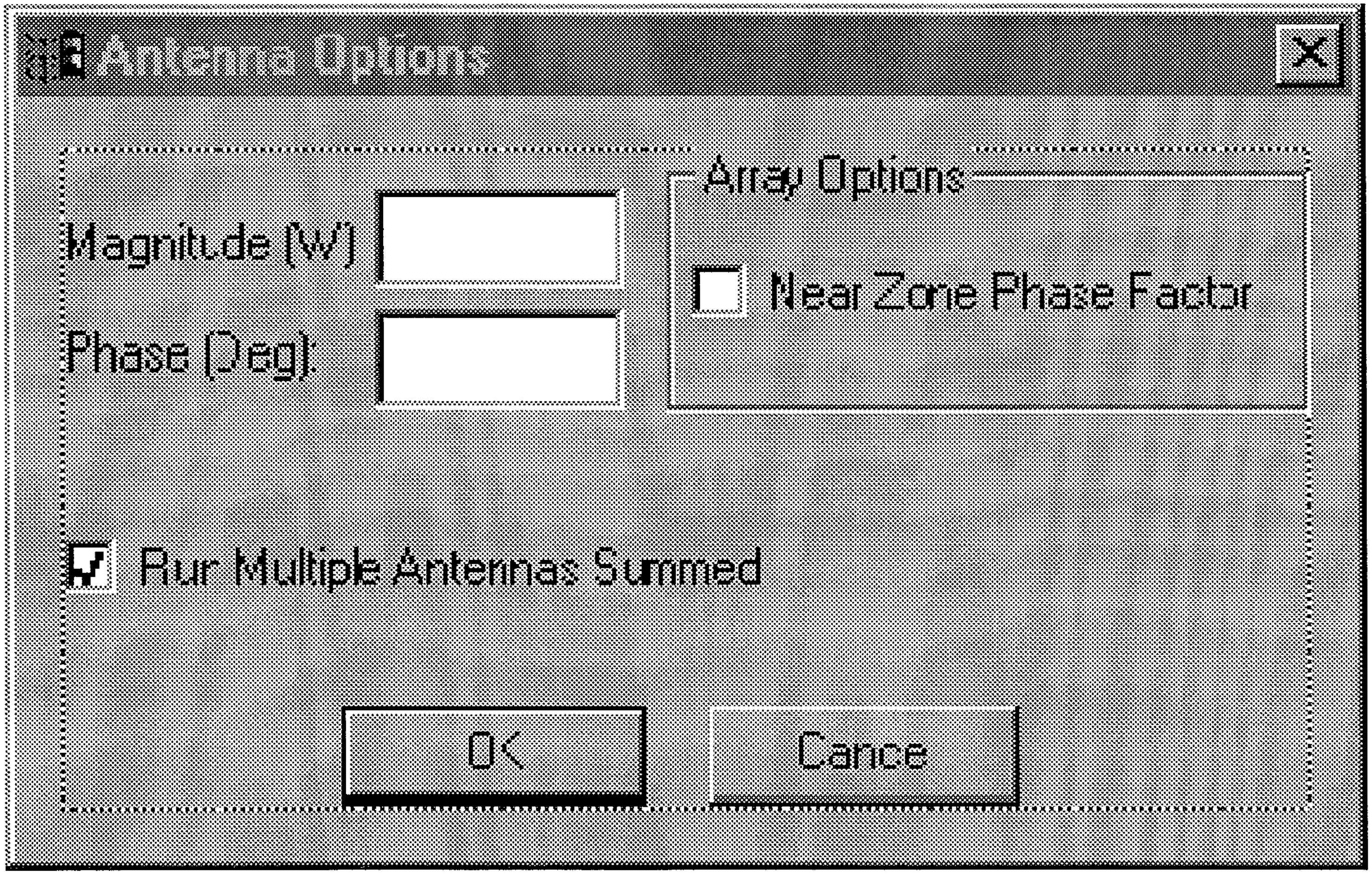


RAY TRACE PLOT - F-15 IN PRIMES (0 TO -35 dB)



WERMS

ANTENNA OPTIONS



Magnitude (W)

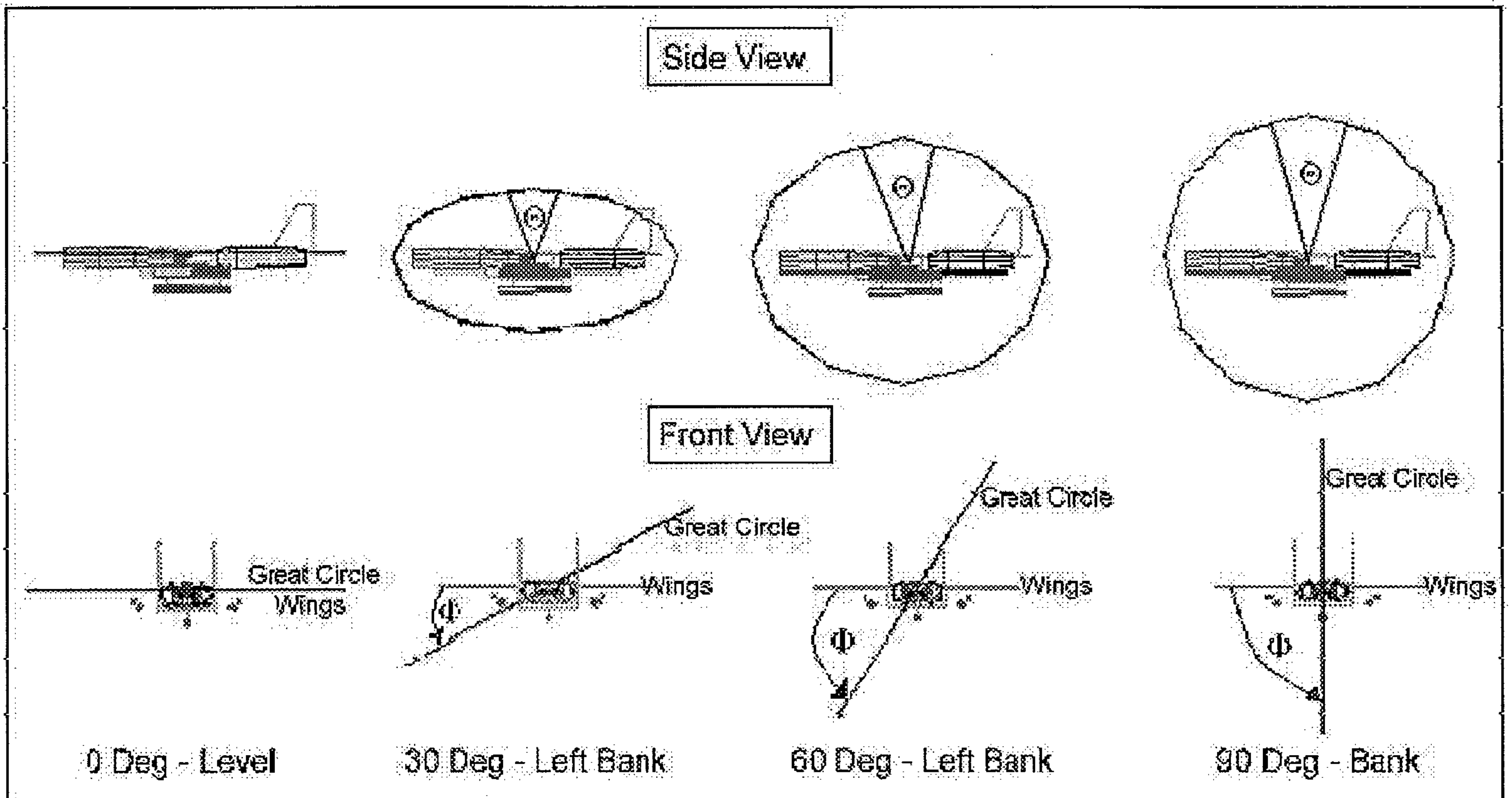
Phase (Deg):

Array Options

□ Near Zone Phase Factor

☑ Run Multiple Antennas Summed

GREAT CIRCLE CUT SELECTION



0 Deg - Level

30 Deg - Left Bank

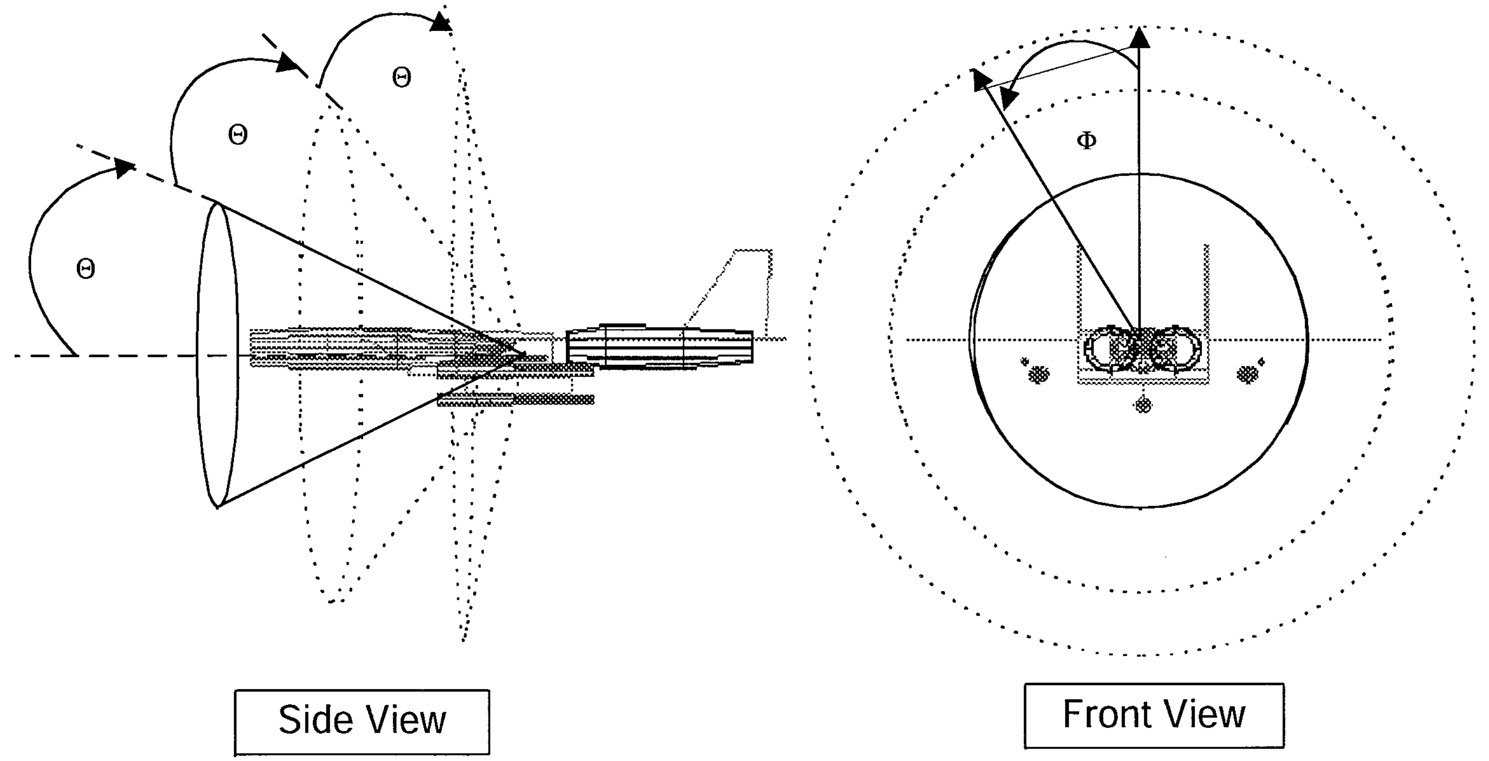
60 Deg - Left Bank

90 Deg - Bank

Phi Step Size = 30 Deg
Theta Increments = 10 Deg

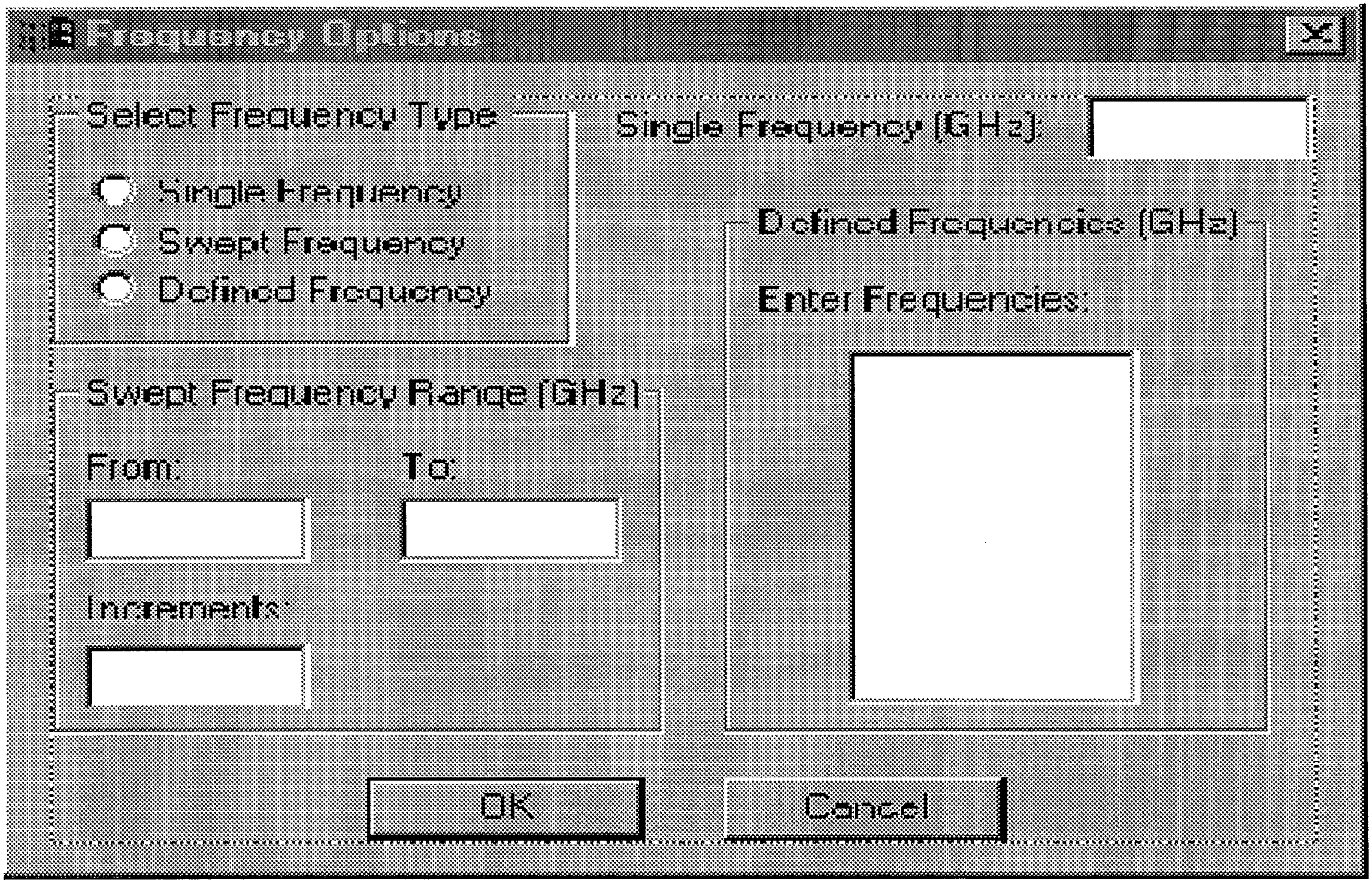
WERMS

CONICAL CUT SELECTION

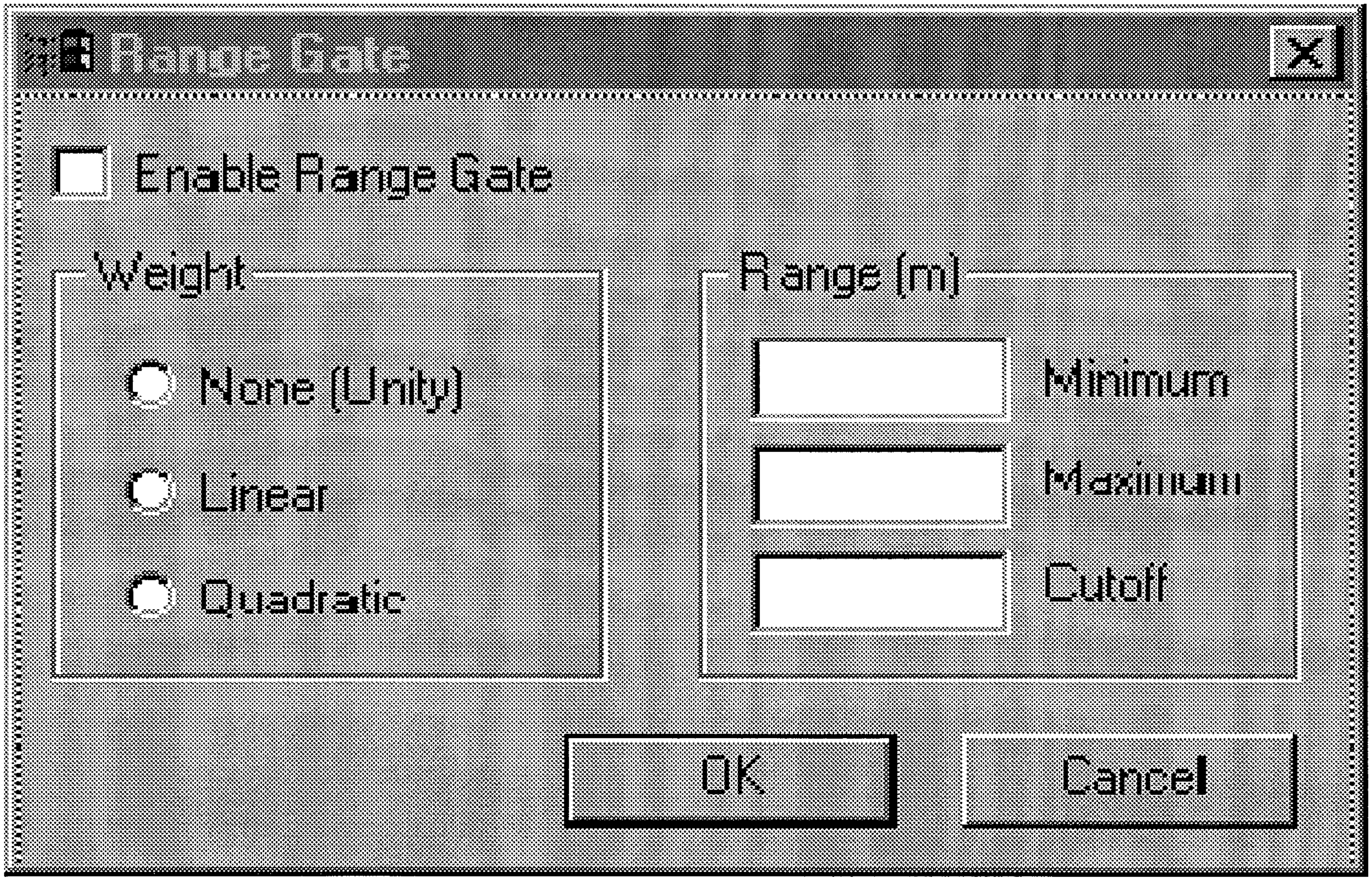


Φ = Phi Increment Angle
Θ = Theta Step Angle

FREQUENCY SELECTION OPTIONS



RANGE GATE OPTIONS



□ Enable Range Gate

Weight

○ None (Unity)

○ Linear

○ Quadratic

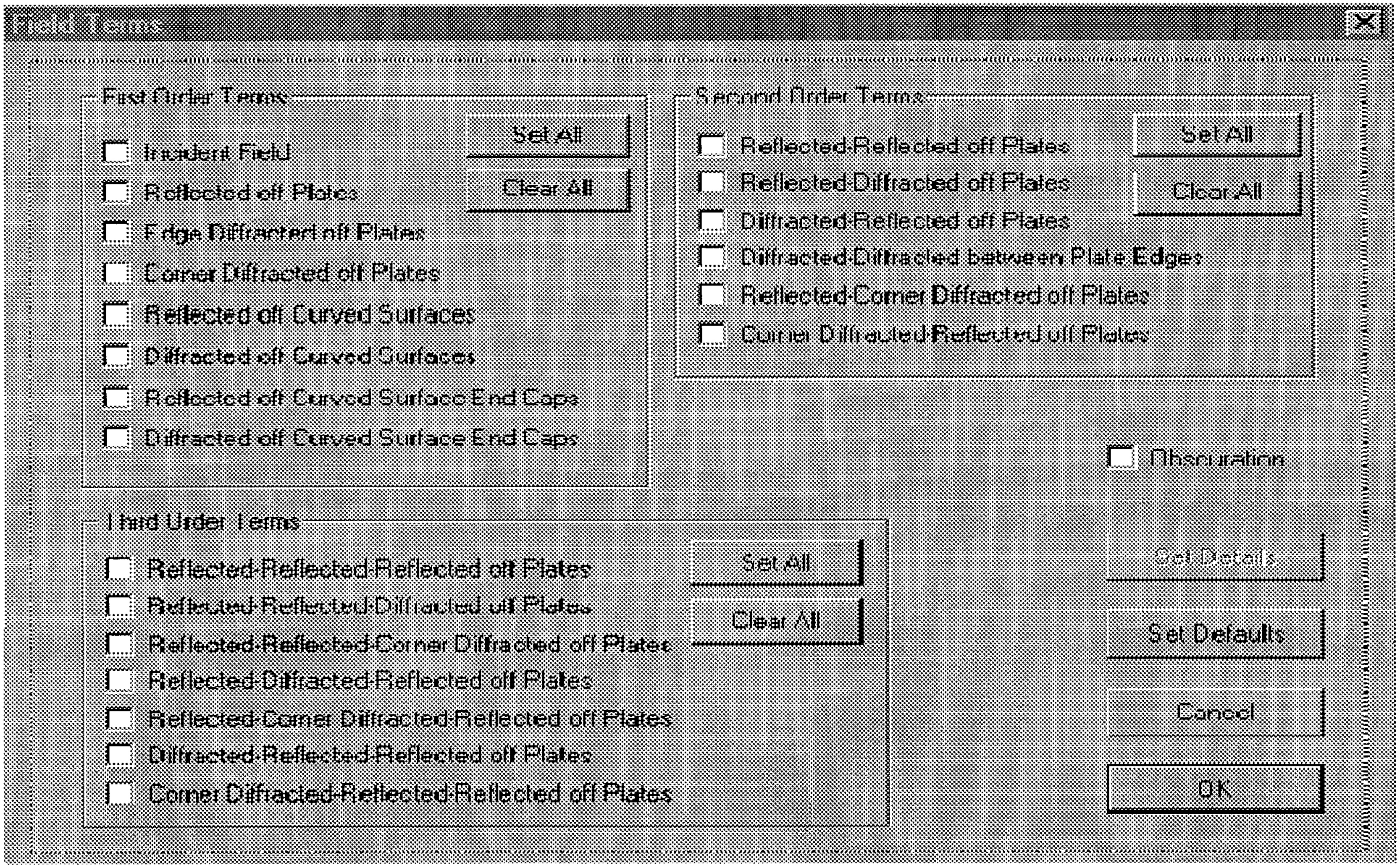
Range (m)

Minimum

Maximum

Cutoff

FIELD TERMS SELECTION



Field Terms

First Order Terms

- □ Incident Field
- □ Reflected off Plates
- □ Edge Diffracted off Plates
- □ Corner Diffracted off Plates
- □ Reflected off Curved Surfaces
- □ Diffracted off Curved Surfaces
- □ Reflected off Curved Surface End Caps
- □ Diffracted off Curved Surface End Caps

Second Order Terms

- □ Reflected-Reflected off Plates
- □ Reflected-Diffracted off Plates
- □ Diffracted-Reflected off Plates
- □ Diffracted-Diffracted between Plate Edges
- □ Reflected-Corner Diffracted off Plates
- □ Corner Diffracted-Reflected off Plates

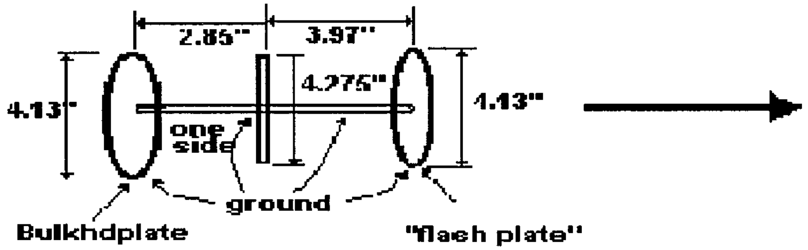
Third Order Terms

- □ Reflected-Reflected-Reflected off Plates
- □ Reflected-Reflected-Diffracted off Plates
- □ Reflected-Reflected-Corner Diffracted off Plates
- □ Reflected-Diffracted-Reflected off Plates
- □ Reflected-Corner Diffracted-Reflected off Plates
- □ Diffracted-Reflected-Reflected off Plates
- □ Corner Diffracted-Reflected-Reflected off Plates

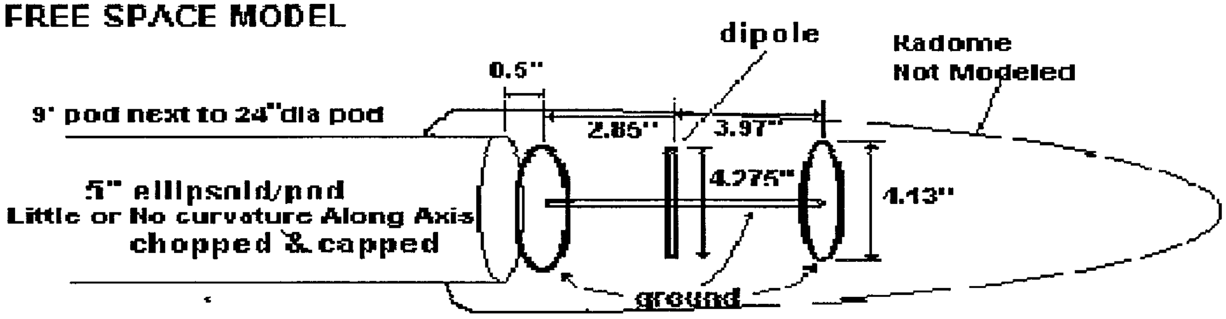
□ Obscuration

WERMS

FLASH-PLATE DIPOLE ANTENNA MODEL

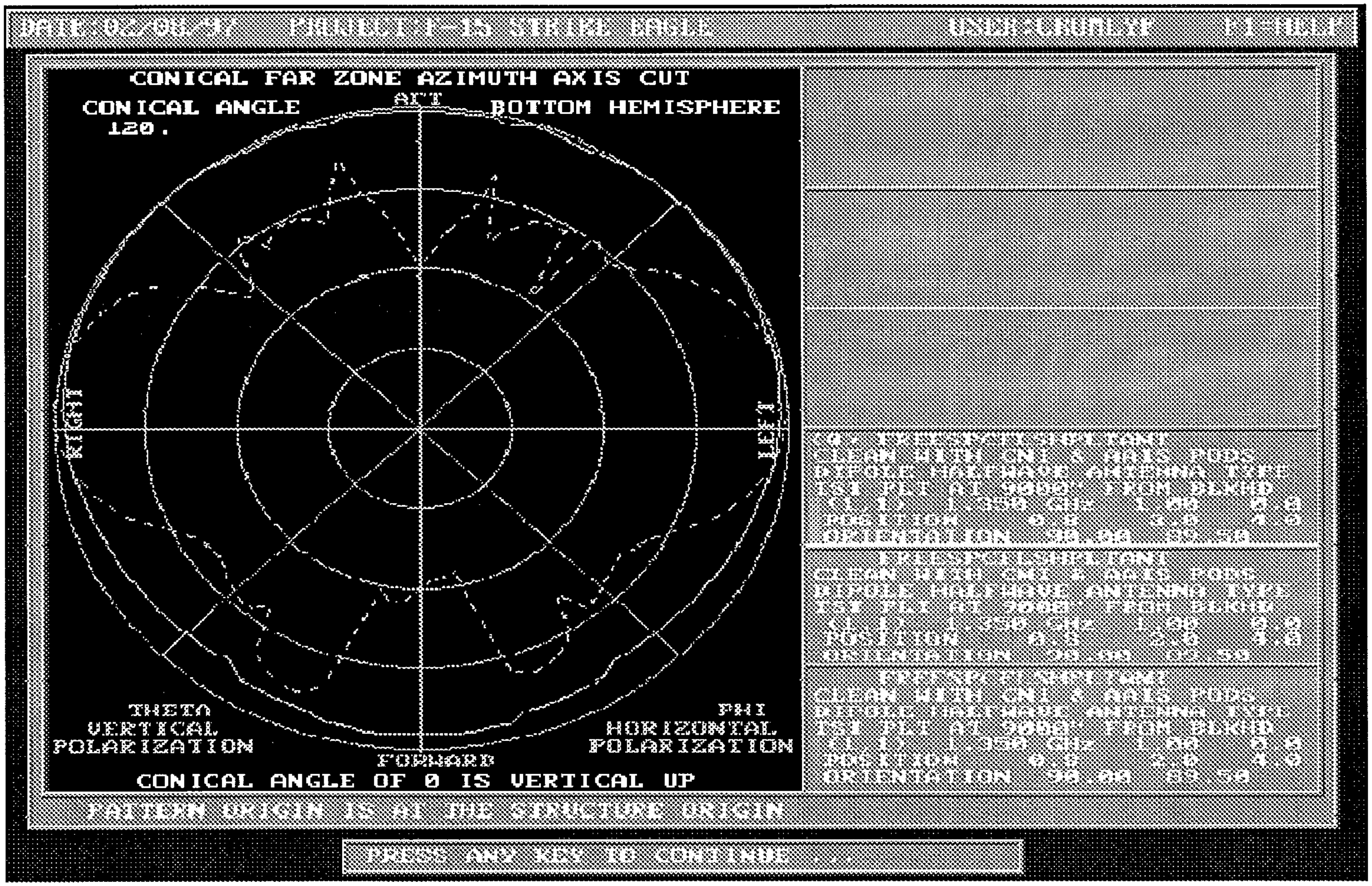


PLATES ARE MODELED OCTAGONAL
FREE SPACE MODEL



ANTENNA INSTALLED ON AAIS POD

FREE SPACE PATTERN for FLASH-PLATE DIPOLE



DATE:02/06/97 PROJECT:F-15 STRIKE EAGLE USER:CRUMLEY F1=HELP

CONICAL FAR ZONE AZIMUTH AXIS CUT

CONICAL ANGLE
120.

BOTTOM HEMISPHERE

THETA
VERTICAL
POLARIZATION

PHI
HORIZONTAL
POLARIZATION

CONICAL ANGLE OF 0 IS VERTICAL UP

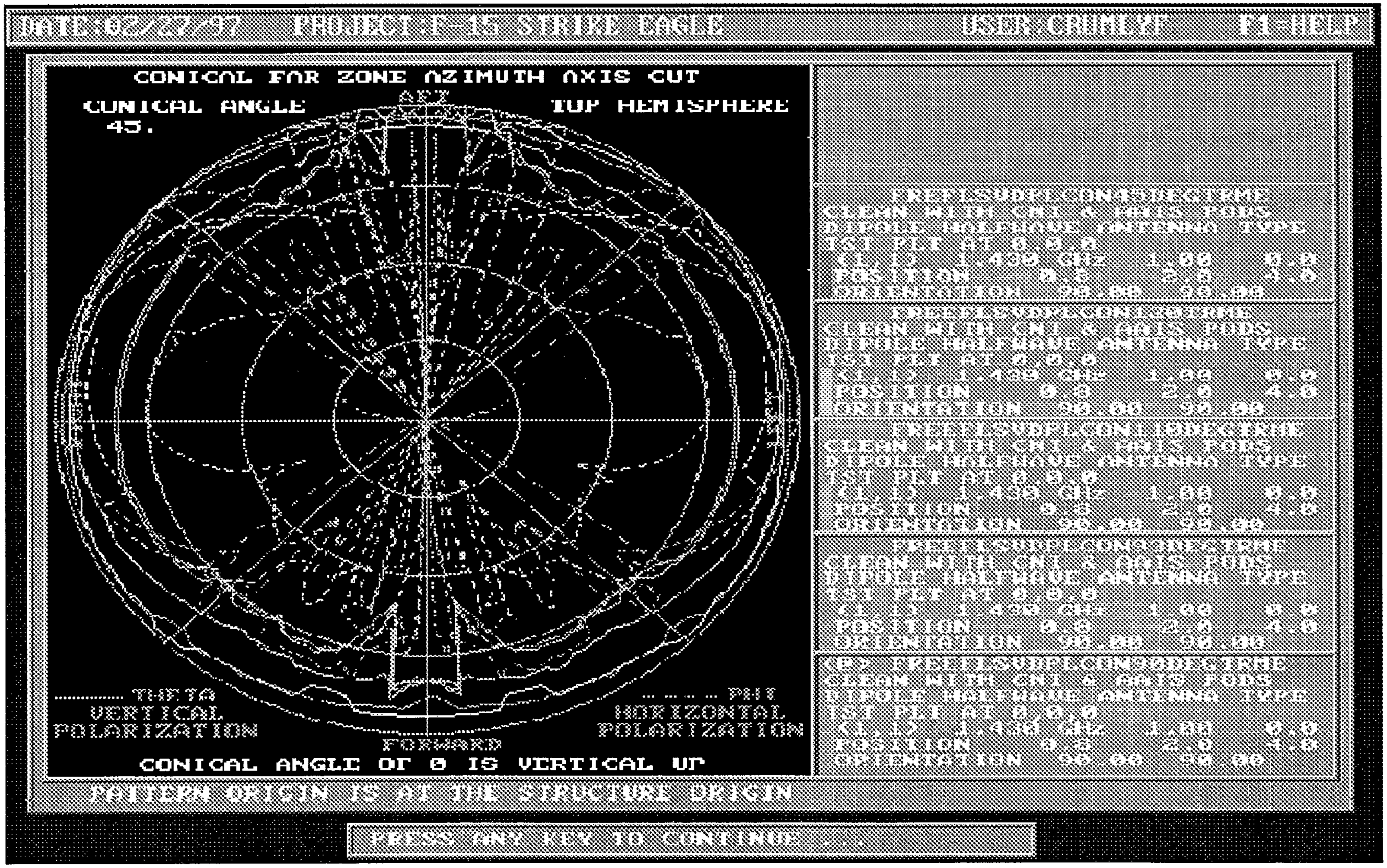
(A) FREESPCFLSHPLTANT
CLEAN WITH CNI & AAIS PODS
DIPOLE HALFWAVE ANTENNA TYPE
1ST PLT AT 9000" FROM BLKHD
(1 1) 1.350 GHZ 1.00 0.0
POSITION 0.0 3.0 4.0
ORIENTATION 90.00 89.50

FREESPCFLSHPLTANT
CLEAN WITH CNI & AAIS PODS
DIPOLE HALFWAVE ANTENNA TYPE
1ST PLT AT 7000" FROM BLKHD
(1 1) 1.350 GHZ 1.00 0.0
POSITION 0.0 3.0 3.0
ORIENTATION 90.00 89.50

FREESPCFLSHPLTANT
CLEAN WITH CNI & AAIS PODS
DIPOLE HALFWAVE ANTENNA TYPE
1ST PLT AT 9000" FROM BLKHD
(1 1) 1.350 GHZ 1.00 0.0
POSITION 0.0 3.0 4.0
ORIENTATION 90.00 89.50

PATTERN ORIGIN IS AT THE STRUCTURE ORIGIN

FREE SPACE VOLUME CUT FLASH-PLATE DIPOLE



DATE:02/27/97 PROJECT:F-15 STRIKE EAGLE USER:CRUMLYF F1=HELP

FREEFLSUBPLCON45DEGTRME
CLEAN WITH CNT & AXIS PODS
DIPOLE HALFWAVE ANTENNA TYPE
1ST PLT AT 0.0.0
X1.12 1.430 GHz 1.00 0.0
POSITION 0.5 5.5 3.5
ORIENTATION 90.00 90.00

FREEFLSUBPLCON[illegible]TRME
CLEAN WITH CNT & AXIS PODS
DIPOLE HALFWAVE ANTENNA TYPE
1ST PLT AT 0.0.0
X1.13 1.430 GHz 1.00 0.0
POSITION 0.0 2.0 4.0
ORIENTATION 90.00 90.00

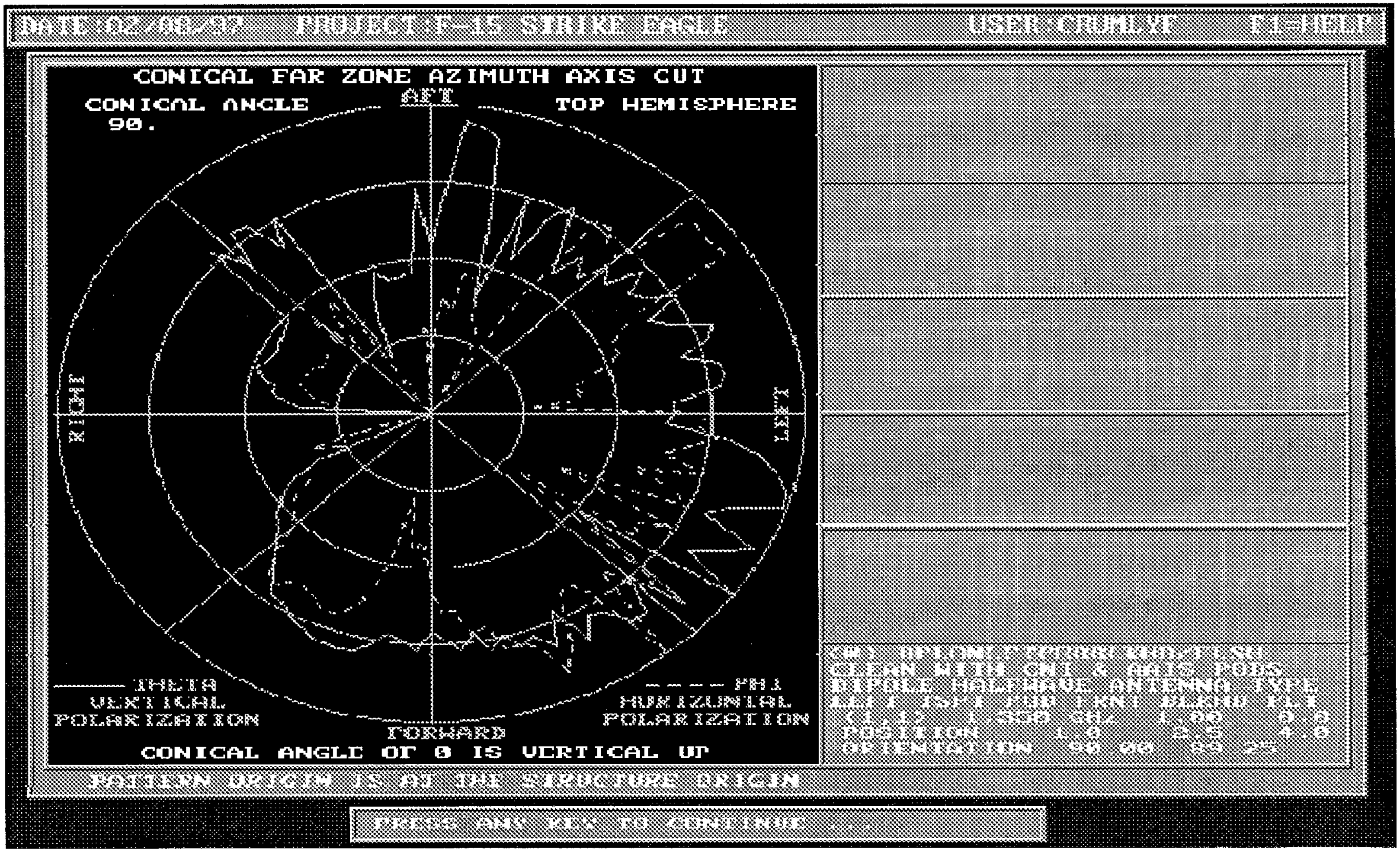
FREEFLSUBPLCON180DEGTRME
CLEAN WITH CNT & AXIS PODS
DIPOLE HALFWAVE ANTENNA TYPE
1ST PLT AT 0.0.0
X1.13 1.430 GHz 1.00 0.0
POSITION 0.0 2.0 4.0
ORIENTATION 90.00 90.00

FREEFLSUBPLCON[illegible]DEGTRME
CLEAN WITH CNT & AXIS PODS
DIPOLE HALFWAVE ANTENNA TYPE
1ST PLT AT 0.0.0
X1.1[illegible] 1.430 GHz 1.00 0.0
POSITION 0.0 2.0 4.0
ORIENTATION 90.00 90.00

<> FREEFLSUBPLCON30DEGTRME
CLEAN WITH CNT & AXIS PODS
DIPOLE HALFWAVE ANTENNA TYPE
1ST PLT AT 0.0.0
X1.12 1.430 GHz 1.00 0.0
POSITION 0.0 2.0 4.0
ORIENTATION 90.00 90.00

PATTERN ORIGIN IS AT THE STRUCTURE ORIGIN

FLASH-PLATE DIPOLE on F-15 LEFT WING AAIS POD



DATE:02/08/97 PROJECT:F-15 STRIKE EAGLE USER:CRUMLYT F1-HELP

CONICAL FAR ZONE AZIMUTH AXIS CUT

CONICAL ANGLE 90.

TOP HEMISPHERE

AFT
RIGHT
LEFT
FORWARD
—— THETA VERTICAL POLARIZATION
– – – – PHI HORIZONTAL POLARIZATION

CONICAL ANGLE OF 0 IS VERTICAL UP

[illegible]
CLEAN WITH CNT & AAIS PODS
DIPOLE HALFWAVE ANTENNA TYPE
LEFT ISPT POD FRNT BLEND FLT
C1 12 1.500 GHZ 1.00 0.0
POSITION 1.0 2.5 4.0
ORIENTATION 90 00 85 25

PATTERN ORIGIN IS AT THE STRUCTURE ORIGIN

PRESS ANY KEY TO CONTINUE

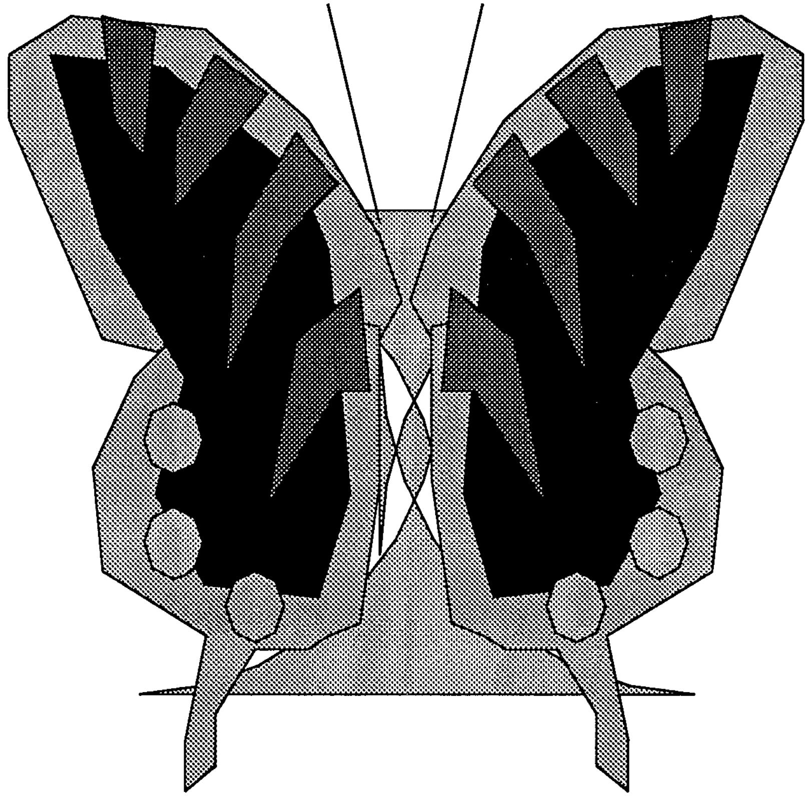
WERMS

WERMS CAPABILITIES

- ANECHOIC CHAMBER CHARACTERIZATION
    - Without and With SUT Installed
- RUN PRETEST SCENARIO BEFORE SUT ARRIVES
- SUPPORT TEST SETUP
- LOCATION OF INTERFERENCE BLOCKING PANELS
- POST TEST ANALYSIS
- ANTENNA PATTERN PREDICTION

THE END

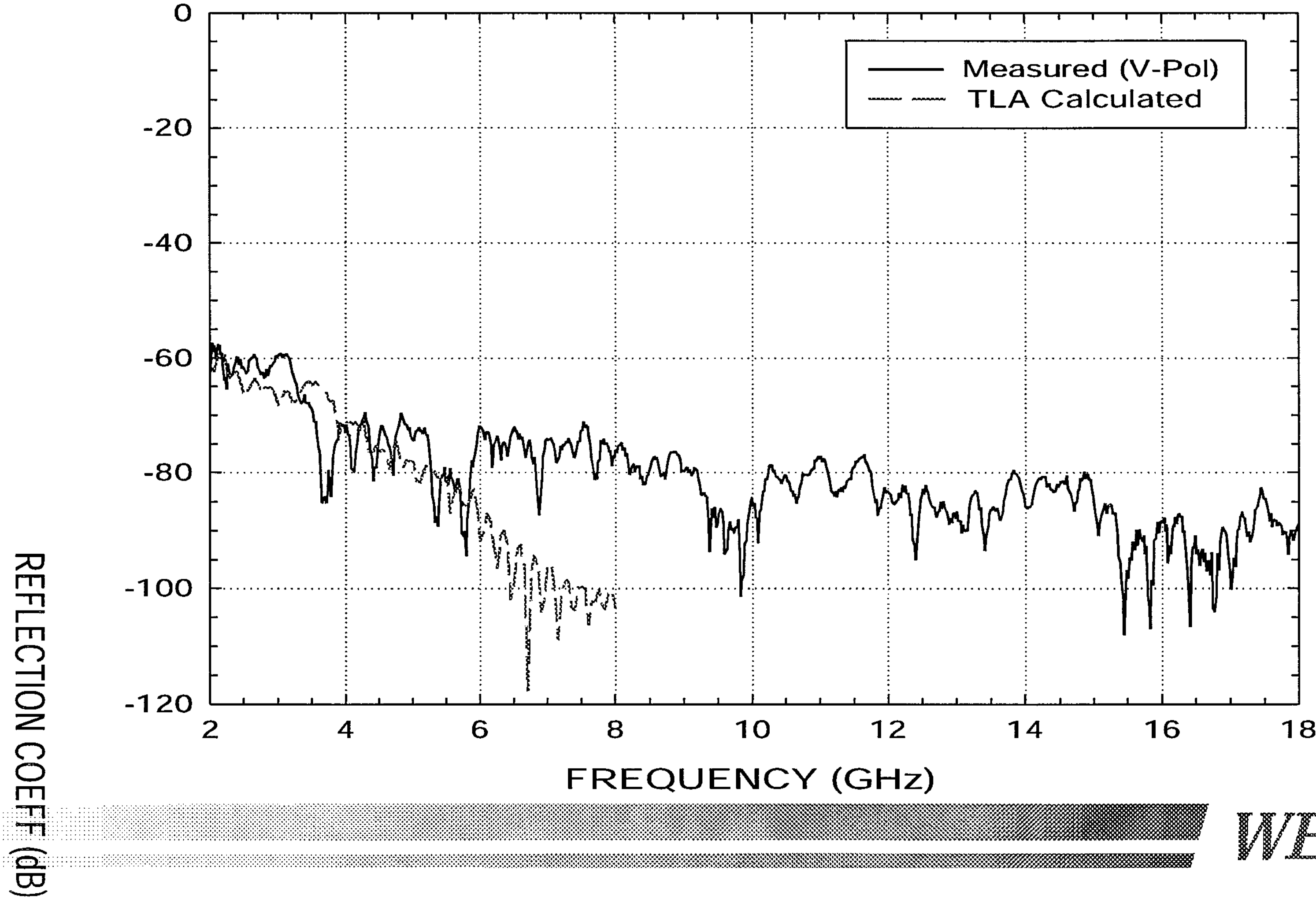
WERMS
HAS
BECOME
A
BUTTERFLY



WERMS

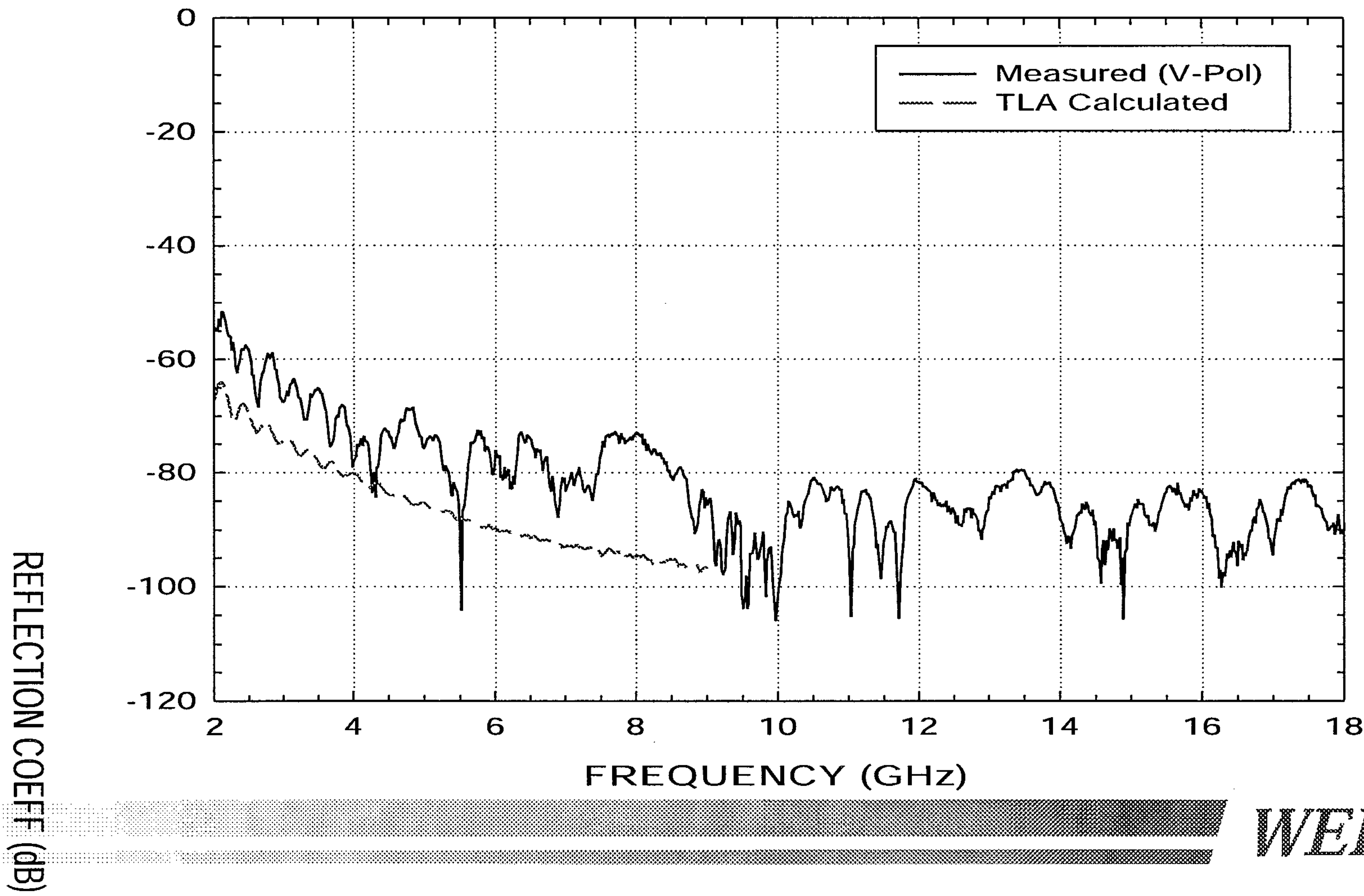
BACKUP SLIDES

REFLECTION COEFFICIENT FOR 24-IN PYRAMID



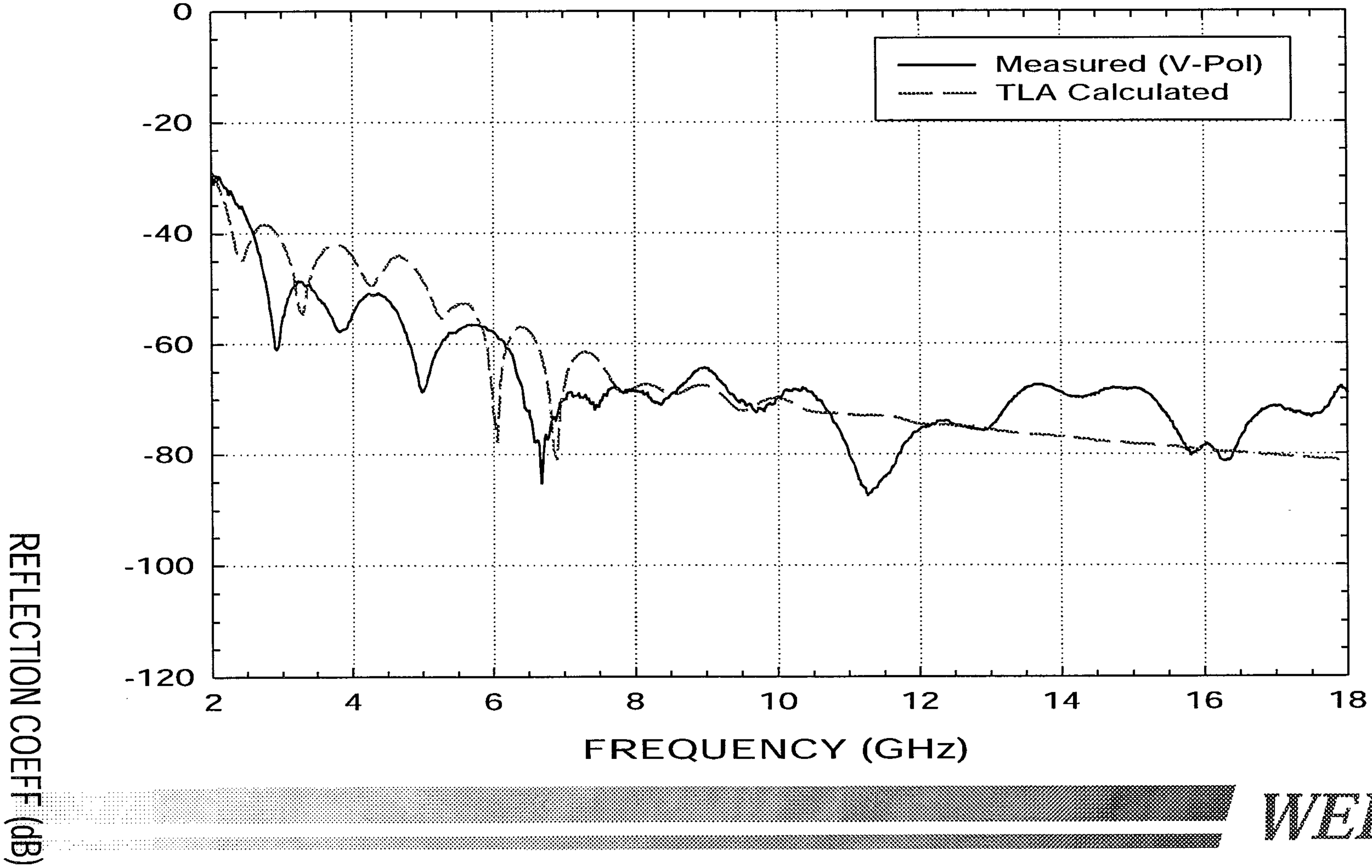
WERMS

REFLECTION COEFICIENT FOR 18-INCH PYRAMID



WERMS

REFLECTION COEFICIENT FOR 5-IN PYRAMID



WERMS

ABSORBER MODELING

- Goal is to provide dominant contributions from pyramidal absorbers to SUT
  - Efficient and compatible with UTD ray tracing
  - Coherent and physically accurate model
  - Good approximation over the bandwidths of interest
  - Easily definable in the GUI
  - Validate model with measurements

ABSORBER MODELING (Continued)

- Emphasize low frequency model, where the absorber less efficient
  - TLA method gives accurate specular reflection coefficients at lower frequencies
  - TLA algorithm is being modified to be compatible with NEC-BSC
  - 24" pyramids are being modeled based on measured dielectric constants

ABSORBER MODELING (Continued)

- Model at higher frequencies
  - Modify NEC-BSC algorithm to be more compatible with TLA algorithm and generalize to account for possible directional behavior
  - Measurements indicate reflection from absorber is nearly constant at -80 dB when TLA gives out
  - Transition point from TLA to higher frequency model will be determined

ABSORBER MODELING (Continued)

- Contribution from near grazing "backscatter" due to reflection from absorber faces added
  - Important at higher frequencies
  - Approximate reflection-diffraction coefficient
  - New class of rays to NEC-BSC added
  - Validated with high resolution measurements

Open Systems in Weapon Systems and the Systems Engineering Process

J. Michael Hanratty

Office of the Undersecretary of Defense (Acquisition and Technology)

Open Systems Joint Task Force

Robert H. Lightsey

Defense Systems Management College

Arvid G. Larson

Walcoff & Associates, Inc.

Abstract

The open system approach is both a technical approach to systems engineering and a preferred business strategy that is becoming widely applied by commercial manufacturers of large complex systems. It has the attention of DoD management who have mandated its use by DoD weapon systems developers. Why? Because without such a change in system development practice, DoD risks being unable to meet the technology needs of the warfighter in the year 2000 and beyond to maintain continued superior combat capability affordably! Open systems designs in both new and legacy weapon systems provide an opportunity towards this end; however, to achieve good open systems designs first demands that a disciplined systems engineering approach be taken to defining the appropriate elements in the system to be opened. This paper discusses the need for a rigorous systems engineering process which incorporates open systems concepts and principles --- where resulting system designs more readily accommodate changing technology to achieve cost, schedule, and performance benefits by promoting multiple sources of supply and technology insertion.

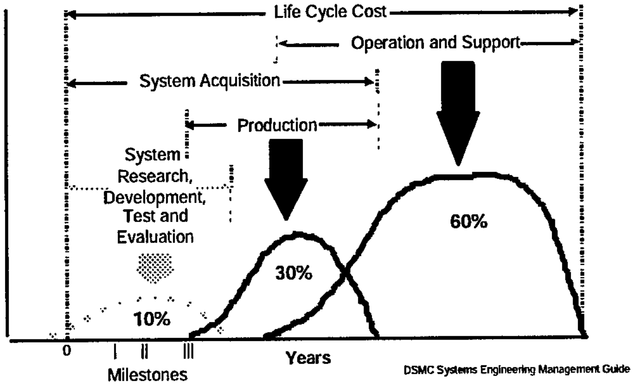
1. Introduction

As both a technical approach to systems engineering and a preferred business strategy for Department of Defense (DoD) weapons systems acquisitions, the open system approach is becoming increasingly attractive to DoD acquisition managers and systems developers. The open systems approach is a major change in weapons system development practice applicable to both new systems design and legacy system upgrade. As a key acquisition reform initiative, this approach perhaps represents the only viable way for DoD to maintain continued superior combat capability affordably.

Today, legacy weapons systems continue to be developed with their own, often unique and frequently closed, infrastructures, making upgrading or modifying them over their expected lifetimes (20 to 40 years) both problematic and expensive. Also, reduced procurement budgets and increased dominance of commercial technology cause acquisition managers to increasingly rely on commercial markets for affordable product development and support. So, as DoD's role shifts from being a technology producer to being a technology consumer, it relies more on commercial products whose design is not controlled by DoD and whose lifetimes are much shorter and more volatile than the weapons systems they support (e.g., years vs. decades). As a result, acquisition managers risk relying on unique products provided by a single supplier at high non-competitive prices and with little opportunity for technology insertion by other suppliers.

2. The Need for an Open Systems Design Approach

An open systems design approach can allow a weapon system program office to achieve and maintain combat superiority in today's challenging acquisition environment. This approach focuses the design process on lowering the entire life-cycle costs (LCC) of weapon systems in contrast to current practice in which a disproportionate focus is placed on the short-term goal of having the lowest development costs. Figure 1 illustrates that 72 percent of LCC are incurred post-IOC during the service lifetime [1]. The ability of the open systems design approach to improve life-cycle supportability is becoming an even more important issue as DoD limits the number of new weapon systems procurements and extends the life of the systems currently fielded.



DSMC Systems Engineering Management Guide

Figure 1. Life Cycle Costs

It seems clear that DoD managers should concentrate on doing things in systems engineering and development that will decrease costs during production and especially during the operations and support (O&S) phase. An open systems approach, basing the weapon system's design on open, commercially supported interface standards with the prospects of a large supplier and customer base, focuses the systems engineering process on developing system designs which consider life-cycle support requirements up front and that support system evolution throughout the system's life.

An open systems approach also mitigates the increased risks of obsolescence due to shortened technology cycle time. Obsolescence risks are significant because technology cycle time, sometimes on the order of months, far outpace weapon system development cycle time, typically 8 to 15 years. By the time a system is fielded, supporting technologies are often outdated -- the US. military cannot afford to be 3 or 4 technological generations behind what is available to the commercial market. Open systems designs, using commercially-supported interface standards permitting upgrade at a relatively low cost, specifically address issues of affordability and supportability associated with long lived system by facilitating evolutionary upgrade with new technology. Generally, this results in superior combat capability over the total system life-cycle, usually at a lower cost to the government.

Another reason that open systems have become so attractive is that DoD is no longer the dominant force in the market place and DoD's procurement budget has been drastically reduced. DoD no longer has the luxury of technology dominance, funded by seemingly unlimited budgets. In prior decades, DoD requirements drove development of new products and new technology. In the today's environment the opposite is true; commercial demand drives product and technology development. However, DoD can now take advantage of commercial innovation, research and development to drive down its cost of developing, acquiring and maintaining weapon systems, leveraging the commercial investment to make the most of available and shrinking defense funds. An open systems approach, using open interfaces supported by commercial and non-developmental components, can substantially facilitate this leveraging.

The bottom-line issue is not only cost: the lives of our servicemen may depend on shortened technology insertion cycle times. In a global market, everyone, including our potential adversaries, will gain increasing access to the same commercial technology base. The military advantage goes to the nation that has the best cycle time to capture the very best commercially available technologies, incorporate them in weapon systems, and get them fielded first. Moreover, since coalition operations with our allies place a high premium on interoperability, it is essential that our systems be compatible and capable of being sustained through a common logistics support structure. Open systems specifications and standards promote standard interfaces and interoperability with our friends and allies.

Each of these many issues will continue to substantively challenge past DoD acquisition practices throughout the foreseeable future. As a result, DoD finds itself with few alternatives but to drastically alter the way it develops, produces and supports its weapon systems. It is neither economically nor technologically feasible to continue traditional closed design approaches. DoD is increasingly compelled to move towards a more open weapon systems design alternative.

3. Open Systems Design Concepts

Simply put, the concept of open systems is a common sense approach that has substantial promise as an approach to meet DoD's continuing need to support systems over increasingly long life cycles in an environment of decreasing resources. In a time when the development of a complex system can span several generations of the faster moving technologies, open system architectures offer the tantalizing prospect of facilitating performance upgrades at affordable costs for the life cycle of the system. The potential and practice of open systems design as an emerging topic within the systems engineering discipline has now been with us for several years. In addition, the use of open systems has received the attention and support of the highest levels of DoD. In 1996, DoD issued a revised directive DoD 5000.2-R that instructs program managers to employ open systems as a design consideration in defense systems engineering [2] . The systems engineering process, with specific reference to the consideration of open systems designs, is integral to achieving the benefits of open systems designs.

While there are many definitions of open systems [3], most have a few characteristics in common. Open systems are those that can be supported by the marketplace, rather than being supported by a single (or limited) set of suppliers, due to the unique aspects of the design chosen. Open systems architectures are achieved by having the design focus on commonly used and widely supported interface standards. One might think in terms of the axle-wheel-tire interfaces employed on commercial cars. By adhering to common standards at the interfaces, the consumer is able to buy tires from a multitude of suppliers, rather than being forced to buy from a single source, as might be the case if the interface characteristics were unique to a single supplier. This ensures costs and quality that are controlled by the forces of competition in the marketplace. Furthermore, the continued support of the system is not subject to the risks associated with having a single supplier go out of business or cease supporting the standard. As the technologies associated with tires change with time, the customer can continue to upgrade and support his vehicle with tires that are built to the accepted industry standard (e.g., conventional sidewall bias-ply technology tires to steel-belted radial-ply technology tires).

However, despite all the high-level attention on open systems, DoD program managers must exercise some care and judgment in their application of the open systems approach. It does not represent a new approach that replaces and makes obsolete previous approaches to engineering complex systems. Moreover, managers should not simply implement an open standard without careful consideration of where (in the system hierarchy) it makes sense to impose standards nor should they simply grasp for a commercial item (CI) solution, whether or not the solution leads to the benefits of open systems architectures. Such actions may encourage program managers to declare that they are achieving open systems attributes, whether or not the system design is well thought out to take full advantage of the benefits that the open systems approach offers. This may give the appearance of achieving open systems architectures but, in fact, such short-sighted decisions work against the long term viability of the system. The open system concept does not replace the need for following a rigorous systems engineering process but, in fact, requires more rigor to ensure that open systems benefits are achieved.

4. Open Systems Applied Within the Systems Engineering Process

Systems engineering is fundamentally a problem solving process that translates needs and requirements as inputs into designs and products as outputs. The systems engineering process typically starts with problem definition as requirements are analyzed. Alternative solutions or system architectures are developed, usually initially through techniques such as functional analysis and data flow analysis. Alternative physical designs are then developed to satisfy the functional or data flows. Trade studies and risk analyses are applied to select a preferred design solution, and that solution is verified against the original requirements.

This process, properly applied, results in a flow down of requirements from the system level to the items below system level. As these requirements are flowed down, the design requirements for the items below system level are defined. Once these lower level design requirements are finalized, the design process proceeds to completion. The result is a design which associates physical entities with the functions that the system must perform, and is consistent with the levels of performance required and with the interfaces specified.

This process, applied without constraints, will lead to the design of a system in which every item is optimized to the requirements in terms of function, performance, and interface. Too often, the results in DoD have been systems that are unique in their designs, which perform their missions quite well, but which require unique equipment and parts to support them, and which can be supported only by a limited set of suppliers. This has historically been a prescription for "closed" systems that are both difficult and costly to support.

The challenge in DoD is to design systems to take advantage of open systems concepts where that makes sense, while continuing to meet the needs and requirements of operational forces. The solution is not to suddenly abandon good systems engineering and simply impose standard interfaces at some point in the system, nor is the answer likely to be found in indiscriminately importing CI solutions into the system architecture. Rather, the real answer is to be found in performing good systems engineering while, as DoD dictates, employing open systems as a design consideration from the outset. The challenge, then, is to integrate systems engineering and open systems design.

To this end, the use of architectures in DoD has become a preferred management approach for implementing an open systems approach [4]. DoD has implemented this concept by defining an interrelated set of architectures: Operational, System, and Technical (illustrated in Figure 2). Basically, the Operational Architecture specifies the "user requirements" which are used as inputs to the systems engineering process to eventually build the weapon system. The Technical Architecture and Product Lines constrain the system's design during the system engineering process. The System Architecture emerges as an output and is constructed to satisfy Operational Architecture requirements within the rules and standards defined in the Technical Architecture. Technical architectures are particularly important to the systems engineering process because they provide the building codes for implementing systems upon which engineering specifications are based, common building blocks are built, and product lines are developed. Note that while, each of these architectures by themselves build nothing, together they provide a management tool which facilitates evolutionary acquisition by supporting insertion of new technology, component reuse, improved weapon systems interoperability and the accommodation of evolving user requirements.

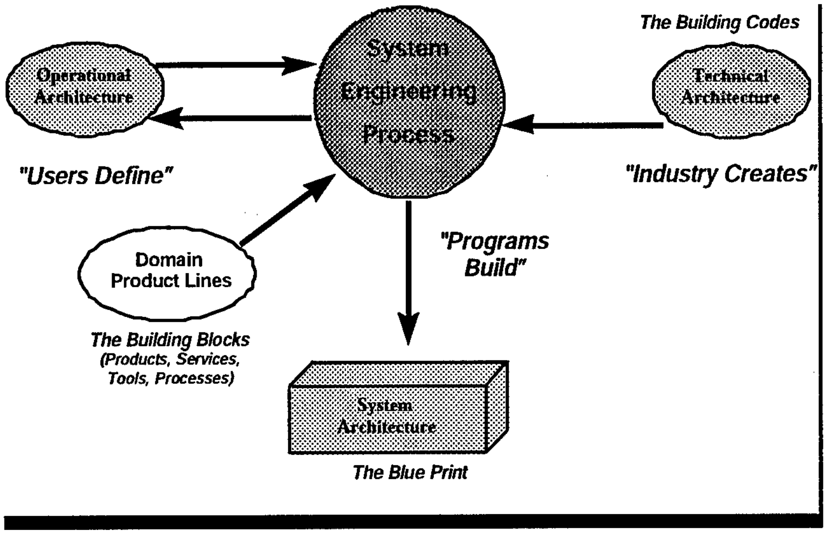


Figure 2. Architectures and the Systems Engineering Process

Who chooses the technical architecture? Does the government choose the architecture; does industry choose the architecture or is the architecture chosen in concert? The government may specify key performance attributes of system building blocks including internal interface standards. However, doing so without adequate input from industry stifles innovation, limits performance and increases cost by attempting to substitute our wisdom for that of the designer. If, on the other hand, we provide no guidance, we may encourage development of proprietary architectures, interfaces and components. That would leave DoD in a position where it must maintain and modify a unique product with a single supplier at a high, non-competitive price. Each program must chose a path between these two extremes. A desirable situation is for there to be a consensus among potential prime contractors and their key suppliers on application of widely accepted standards.

Using an open systems approach to the systems engineering process helps achieve an integrated design solution which is resilient to changes in technology throughout the life of the system. Open systems (engineering) achieves this resiliency in "life-cycle supportability" by engineering systems according to the following principles and practices (illustrated in Figure 3):

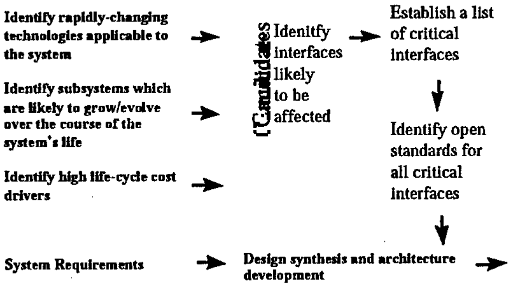

Identify rapidly-changing technologies applicable to the system
Identify subsystems which are likely to grow/evolve over the course of the system's life
Identify high life-cycle cost drivers
(Candidates
Idenitfy interfaces likely to be affected
Establish a list of critical interfaces
Identify open standards for all critical interfaces
System Requirements
Design synthesis and architecture development

Figure 3. Open Systems Analysis for Integrated Design Solution

- Identify as critical the interfaces to subsystems or components which are likely to change due to their dependence on rapidly evolving technology, are likely to have increasing requirements, have high replacement frequency or have high costs. Such components present both the highest obsolescence risks and the greatest opportunity for future technology insertion.
- Use open standards for these critical interfaces that are supported by the broader community, , are considerate of life-cycle support requirements, permit evolution with advances in technology, and support technology insertion.
- Use a modular design approach combined with well defined standards-based interfaces among modules to isolate the effects of change in evolving systems, serving to reduce the need for redesign as the system is upgraded.
- Identify the lowest level at which the government maintains control over the interface standard, and anticipate how this level may change over time. Below this level the contractor is permitted to use its best, perhaps proprietary, practices to improve or discriminate its product in the marketplace.
- Verify all performance requirements and reevaluate their stringency. Reallocate requirements as necessary to permit the wider use of open standards throughout the system.
- Implement consistent conformance management practices to ensure that products procured for the system conform to the established profile so as to prevent being limited to one supplier who might unilaterally extend that interface.

The key to achieving the benefits of open systems designs lies in making open systems an integral part of the classic systems engineering process and in applying open systems at all stages of the product life cycle. The open systems approach to design will never replace or make obsolete that process -- if anything, it demands that the process be even more rigorously applied. As is illustrated in Figure 4, each of the major aspects of the systems engineering process must include consideration of open systems design concepts and principles:

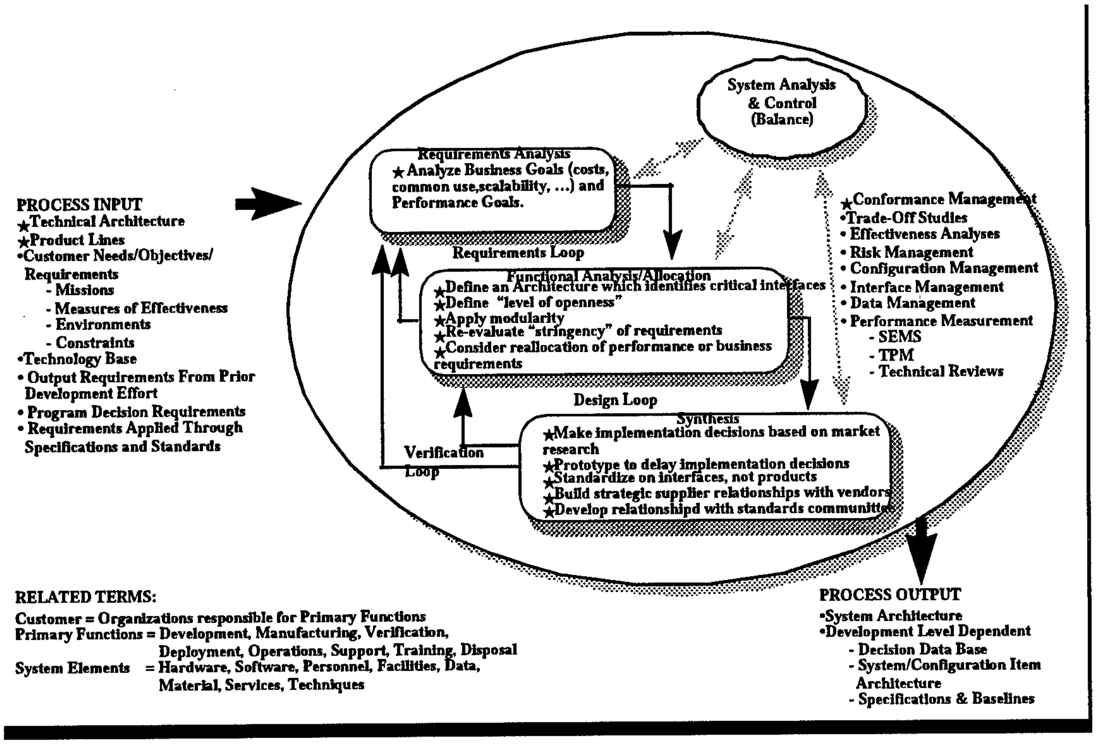

PROCESS INPUT
★Technical Architecture
★Product Lines
•Customer Needs/Objectives/ Requirements
- Missions
- Measures of Effectiveness
- Environments
- Constraints
•Technology Base
• Output Requirements From Prior Development Effort
• Program Decision Requirements
• Requirements Applied Through Specifications and Standards
System Analysis & Control (Balance)
Requirements Analysis
★Analyze Business Goals (costs, common use,scalability, ...) and Performance Goals.
Requirements Loop
Functional Analysis/Allocation
★Define an Architecture which identifies critical interfaces
★Define "level of openness"
★Apply modularity
★Re-evaluate "stringency" of requirements
★Consider reallocation of performance or business requirements
Design Loop
Synthesis
★Make implementation decisions based on market research
★Prototype to delay implementation decisions
★Standardize on interfaces, not products
★Build strategic supplier relationships with vendors
★Develop relationshipd with standards communitties
Verification Loop
★Conformance Management
•Trade-Off Studies
• Effectiveness Analyses
• Risk Management
• Configuration Management
• Interface Management
• Data Management
• Performance Measurement
- SEMS
- TPM
- Technical Reviews
RELATED TERMS:
Customer = Organizations responsible for Primary Functions
Primary Functions = Development, Manufacturing, Verification, Deployment, Operations, Support, Training, Disposal
System Elements = Hardware, Software, Personnel, Facilities, Data, Material, Services, Techniques
PROCESS OUTPUT
•System Architecture
•Development Level Dependent
- Decision Data Base
- System/Configuration Item Architecture
- Specifications & Baselines

Figure 4. Integrating Open Systems and the Systems Engineering Process

Requirements analysis must emphasize the balancing of business goals (costs, common use, life cycle supportability, etc.) with technical goals (functionality, performance, interfaces, and other constraints). As the systems engineering process iterates, the requirements analysis step is revisited to consider cost-performance tradeoffs to meet most performance objectives while achieving as large as possible reductions in life-cycle costs. The stringency of requirements is reevaluated to consider the use of open standards for interfaces as performance requirements are balanced (weighed) against business requirements. To do this, engineers need to be better trained to incorporate life cycle cost in design and provided tools which allow them to rapidly assess life cycle cost impacts. Under any circumstances, users have a requirement for systems that are supportable and affordable, and these requirements demand that one consider open architectures as system elements are defined.

Functional analysis and allocation must define an architecture which provides a framework for identifying interfaces critical to achieving system business and technical performance goals. Requirements should be allocated with a view toward achieving functional modularity. Functional modularity can facilitate physical modularity and the use of open interfaces to support system evolution goals. As the systems engineering process iterates, this step is revisited to allocate functionality so as to modularize those components or subsystems which are dependent on rapidly evolving technology, have high replacement frequency or are high costs and to reallocate performance or business requirements as necessary to allow for the use of open interface standards during synthesis.

Synthesis and design should continue the search for alternative system architectures that will satisfy requirements. To be effective, good design synthesis demands an iterative approach that involves revisiting the functional allocations and developing alternative physical solutions until a balanced design (in terms of cost, performance, and risk) is achieved. Modularity should be used in system design where interfaces between modules are based on open, widely-supported interface standards. Modularity should be based on well-defined interfaces to isolate components that are likely to change over time (e.g., dependent on rapidly evolving technology, have high replacement frequency) or are high costs since these components present the highest obsolescence risks and the greatest opportunity for future technology insertion. Well-defined interfaces are used to decouple system components and define firewalls to contain evolution of lower level component upgrades and modifications, thereby minimizing future redesign, and possibly retesting, when components are upgraded. In addition, physical modularity should be aligned with functional partitioning to facilitate the replacement of specific subsystems and components without impacting others.

Design iteration should sequentially reconsider the allocations of function and performance that define the design requirements for each system component with the objective of achieving user (customer) requirements within an optimal open systems solution. From an open systems perspective, if this sequential iteration is stopped as soon as the first acceptable technical solution is achieved, there are two probable results: either the solution will be shown to (1) require unique designs that require new development, or (2) an open solution, if imposed at this point, will likely not meet all the requirements of the user. However, in most cases, a final design can almost certainly be developed that results in system architectures that include some items that are "open" and other elements that are not. Although open designs are the objective, it is neither necessary nor in some cases even possible that every element or item of most complex systems be totally open.

Systems analysis and control must include conformance management incorporating both implementation and applications conformance testing. The selected conformance approach must be fully defined and documented so that it is understood by all parties. The degree to which open systems benefits can be achieved will depend largely on how well the product design conforms to selected standards. Completely defined interface profiles will allow vendors to build standards-based components and allow users to design systems to use standards-based components. In all cases, candidate components should be tested against detailed system profiles to ensure that components conform to profiles.

5. Open System Design Challenges

The open approach to system design offers considerable benefits, already discussed, in terms of life cycle support, affordability and timely technology insertion. The approach also carries with it some substantial differences in the way that systems will be managed and supported. Since by its nature open systems designs will involve increased use of commercial and non-developmental items in systems architectures, the government will necessarily have to plan for significant differences in the way systems are managed from a technical perspective. These differences cut across almost every aspect of engineering management, and while space prohibits an exhaustive treatment, examples include the following:

- Standards based architectures lessen the degree of control that DoD can expect to exert. Changes, fixes, and updates will likely be under the vendor's control. This can have a significant impact on system support.
- Standards based elements of the architecture are likely to be faster and cheaper to acquire than a comparable developmental item but may take more time to integrate and test.
- Standards selection is risky. Acquisition will require substantially more knowledge of the current state of the art and the marketplace on the part of the government.
- Standards evolve with time. It is difficult to project the extent to which a given standard will endure. It's equally challenging to determine when to move from one standard to the next.
- Standards based architectures tend to change the focus of systems engineering from design to integration. The challenge is to achieve performance requirements without detailed control over the component design specification.
- An item, once integrated, may affect other system parameters. Commercial and NDI items make testing an on-going and continuing activity to verify that items can integrate successfully into systems.
- The use of commercial and NDI requires that support concepts be developed early in the acquisition cycle.

While this is hardly an exhaustive list, it makes the point that open systems engineering introduces new issues into the management of the technical aspects of programs. There are many potential benefits, but, likewise, there are challenges and problems that the manager must be alert to anticipate and overcome.

6. Summary

The objective of open systems acquisitions is to provide the warfighter the most effective weapon systems possible. An open systems approach to systems engineering facilitates this throughout the life of the system. Open systems designs provide an opportunity to achieve affordable designs which can more readily accommodate changing technology while promoting multiple sources of supply; however, to achieve good open systems designs first demands that a disciplined systems engineering approach be taken to defining the appropriate elements in the system to be opened.

Most systems will not be completely open in their architectures, but a well-engineered design will result in a design strategy that takes maximum advantage of the benefits available from opening the design. Associated with an open approach is the need to focus on and manage the interfaces between open system elements and other elements of the system. Choosing well-known and accepted industry standards and applying them in a controlled manner will go far toward achieving the desired results. Overall, the system architecture resulting from a system engineering process should be linked to a business case analysis. Architecture decisions should be traceable to performance, life-cycle cost, schedule, and risk. The alternatives for support, maintenance and upgrade should be evaluated.

For maximum benefit, an open systems approach should focus on planned use of designs across a system or domain. As designs are opened, managers must be aware of the fact that support and acquisition strategies will necessarily be impacted. These impacts must be anticipated and planned for from the outset during system design.

More open systems information and reference materials are available on the Open Systems Joint Task Force home page on the Worldwide Web at http://www.acq.osd.mil/osjtf

References

1. Systems Engineering Management Guide, Defense Systems Management College, Fort Belvoir, VA (1990), DTIC #ADA 223-168, see Chapter 17, Life Cycle Costs.

2. DoD Regulation 5000.2-R, Change 2, 6 October 1997, see Paragraph 4.3.4, Open System Design.

3. Final Report of the Tri-Service Open Systems Architecture Working Group, Department of the Navy, Office of the Assistant Secretary (RD&A), 2 September 1993, see Appendix B, Definitions. Also see definitions on the OS-JTF home page at http://www.acq.osd.mil/osjtf.

4. Implementation of the DoD Joint Technical Architecture, Memorandum dated 22 August 1996, issued jointly by the Under Secretary of Defense (A&T) and the Assistant Secretary of Defense (C3I).

Lessons Learned Using COTS in Real-Time Embedded Systems

Presented to the Joint Avionics and Weapons Systems Support Software and Simulation Conference

June 1998

Robert Rosenberg
Sabre Systems, Inc.
Pine Hill Technology Park
48015 Pine Hill Run Road, Building C
Lexington Park, MD 20653

Lessons Learned Using COTS in Real-Time Embedded Systems

Presenter: Robert Rosenberg, Sabre Systems, Inc.

Track: SDT1

It is certainly not news that the use of commercially-available-off-the-shelf (COTS) hardware and software is revolutionizing the acquisition process for new real-time embedded systems. Designs using COTS hardware and software can be the only approach to meeting today's aggressive schedule and cost budgets, while lowering production risks. Using Open Systems is an attractive approach to inexpensively incorporate increased performance as technology improves. It is hard to imagine a program manager who would not want to try to take advantage of these benefits.

DOD is not only encouraging, but also mandating the use of COTS hardware and software in the development of new and replacement (sub)systems to meet current budget constraints.

However, the use of COTS must be carefully managed or the results of these benefits will disappear into a life cycle maintenance nightmare. There are several lessons that have been learned about the use of COTS that should be considered before planning a new program.

1.0 Introduction

Historically there were sound reasons not to use COTS. Today technology has evolved and many of the historic problems have been overcome.

The use of Commercial-Off the Shelf (COTS) hardware and software in real-time systems is not new. I have been involved in developments using COTS in real-time systems since 1983 and I was hardly a pioneer. The use of COTS in real-time systems in the past was limited by technical considerations, operational suitability issues, and life cycle concerns. The lower cost of COTS was always attractive, but for most programs these limitations made its use impractical.

The major performance hurdle to using COTS in Command and Control (C2) real-time applications was always that COTS hardware and its shrink-wrapped operating systems were not designed to handle the quantity of interrupts that C2 systems need to process in a timely fashion. As technology developed, it was possible to use intelligent controller cards to solve this technical issue. There were only a handful of vendors who started to design their products for the real-time market. Today there is a wider range of hardware and operating system products available from a growing number of vendors whose systems handle frequent interrupts in a fashion that will support demanding real-time requirements.

In the past, COTS was designed to operate in a temperature and humidity controlled environment. That made it unsuitable for many real-time applications, which were required to operate reliably in more extreme environments. Today many hardware manufacturers are designing their equipment to be more tolerant of environmental conditions with improved reliability.

One of the largest management reasons for not using COTS was the lack of control of parts availability. If a program manager (PM) was willing to pay to operate an assembly line, he could guaranty the continued availability of spare parts. When it was necessary to close a line, that PM could purchase a life time supply of spares before closing the line. When using COTS, programs could try to obtain a commitment from the manufacturer to provide notice before making a part unavailable, but I can recall more than one case when such an agreement was in place and the vendor either failed to honor it or forgot about it. Parts obsolescence issues remain one of the largest challenges in the use of COTS.

Another reason that COTS was not used in the past was economic. Programs did not want to incur the expense to re-host and maintain standard software that was already hosted in a MIL-SPEC computer. Sometimes, it was less expensive to purchase an identical MIL-SPEC computer to re-host that function. The C2P hosted on an UYK-43 processor that decoded Link 11 and Link 16 messages was a good example of a function that did not make economic sense to re-host, several years ago. Software reuse is still a sound approach to saving money and is one alternative that should still be considered when designing new systems.

Sometimes it made little sense to re-host software on COTS if a program already had a design tightly coupled to an existing processor. If that processor became obsolete and if its replacement had a compatibility mode, it became very inexpensive to re-host on the replacement MIL-SPEC computer. Many UYK-7 applications were re-hosted on UYK-43s for that reason. Today's faster processors require less tightly coupled designs to meet performance requirements.

Sometimes the use of COTS in the past was not considered for other reasons. For example, Congress mandated the use of the UYS-2 common signal processor on several programs. Other times it was prudent for a program manager to use a "MIL-standard" computer to widen the logistics base for those processors and take advantage of their existing logistics infrastructure. Top level policy has changed and the use of "MIL-standard" computers is not required or encouraged for new applications. The concept of controlling logistics costs is still valid and can be applied to COTS systems.

Since most of these reasons are no longer valid, the use of COTS is being driven by an effort to substantially reduce development cost and reduce the time required to develop new systems by leveraging off of the extremely large commercial base of COTS hardware and software development tools.

This paper presents lessons derived from my experience related to the use of COTS and how it can be successfully used in real-time systems. It is presented primarily from a government program manager's point of view, but provides information useful to others involved in developing real-time systems. It only addresses acquisition initiatives, policies, development techniques, and practices to the extent that they are uniquely applied to COTS procurement and development. The references are not the source of the information presented; they have been included to provide the reader with additional background on the many topics discussed.

2.0 Methods Used to Reduce the Risk of Initial COTS Developments

The development of new systems using COTS can be successful if the proper approach is taken.

To utilize COTS hardware and software, changes are required to the way that real-time systems have been traditionally developed. These changes affect the entire range of the project and include both the buyer and developer. To achieve an accelerated low cost development, it is necessary to modify the approach to system acquisition from system specification through design, testing, acceptance and maintenance.

Successful COTS developments require a new mind set for both the buyer and the systems integrator[i]. In the past the buyer defined requirements and then ensured that the developer properly designed, built and tested a system to meet those requirements. The developer closely held many of the design tradeoffs and cost issues (in fixed price contracts). For COTS developments, roles need to be redefined. For systems that consist entirely of COTS hardware, the role of the developer is closer to that of a "system integrator". For systems that consist of a mixture of COTS and custom hardware, the roles need to be appropriately tailored.

Buyers and system integrators need to work closely to consider the cost and technical tradeoffs. Many system developers have a traditional business base developing custom hardware. Those organizations need to overcome relying on those skills and need to become more adept at doing market surveys and evaluating existing products. A successful COTS development will define the developer's role in many respects as a system integrator providing services. Product responsibility should be defined as a joint buyer and system integrator responsibility.

It is critical that some of the traditional boundaries between buyer and developer be reformed into a closer team approach. One example of this type of team currently in use in many projects is the Integrated Product/Process Team (IPT). By working together, the buyer and developer can work the technical tradeoffs to obtain the best affordable technical solution. However, to make IPTs work effectively, both buyer and developer need to modify their methods of system development. Initial training in the planned team organization can be held separately prior to contract award. After source selection, additional training should be held jointly to help form an effective team. Periodically during project execution, each IPT should take some time out from the daily issues to evaluate how well the team is operating and how it might improve.[ii]

There is always resistance to using new methods. Most of us tend to rely on methods that have been successful for us in the past. To overcome this natural tendency, there needs to be incentives for the buyer and developer to cooperate. Proper use of Award Fee and Incentive Fees usually provide a substantial incentive for developers. One aspect of each award fee period should be an evaluation of each IPT's operation as a team. Spot awards are an effective way to incentivize members of the buyer's team to reward teamwork.

Lessons Learned:

1. **Customize your acquisition approach. Make it clear to all parties that it is a team effort. Make it to everyone's benefit to work as a team.**

2. **Define ways to reward those team members who find innovative cost effective solutions.**

2.1 Modifying the Approach to Defining Requirements

A flexible approach to defining system requirements is necessary to take full advantage of COTS in a system's development.

Traditional system developments have defined acceptance criteria and then have provided those requirements to one or more vendors to propose the "best" design to meet the requirements. Cost and schedule were usually a key factor in determining the "best" value. However, this approach was flawed since sometimes a small change in a requirement value could cause a large increase in cost.

This is especially true when using COTS. A small change in a requirement can eliminate all available products. A good example of this is operating temperature. Many COTS vendors design and qualify their products to 50 degrees C. Many traditional real-time systems required operating temperatures to 55 degrees C. Since many requirements define a standard of suitability, it is necessary to consider whether that standard is absolute or whether a minor variation would be acceptable.

There are many other examples of requirements that are quantified to establish acceptance criteria, but the specific value is not absolute. Examples of these items are Reliability (MTBF), Fault detection, Fault isolation etc. These values are specified because we want to buy systems that are reliable and maintainable. However, one of those values can be adjusted slightly to achieve significant cost savings.

There are several ways to avoid inadvertently driving cost. One of the most frequently used methods for reducing costs is the use of draft Request for Proposals (RFPs) and systems specifications. This process is very good for avoiding the most obvious problems early in the development cycle, but usually does not identify all of the issues. Other cost drivers may be identified later as the developer's design matures and he is able to identify other factors that reduce the size of the candidate hardware domain.

The use of Alpha Contracting is also a good way of early identification of cost driving requirements[iii]. Alpha Contracting is a process where the buyer and system integrator work together to write the statement of work and specification to define a product that meets the buyer's needs without inadvertently adding cost drivers.

An innovative way for dealing with cost driving requirements is the use of placeholder requirements. A fair competition can be held since all vendors are required to bid to the place holder requirements' value. After the contract is awarded, the developer is required to study each placeholder value to determine whether small reductions in any of the values can result in cost savings. The buyer can pool savings to deal with contract changes and the developer will eagerly participate if the identification of these savings results in incentive fees. My experience has been that very few of the requirements actually change. After considering all of the candidates, both buyer and system integrator subsequently agreed that only a few changes to the original placeholder requirements were desirable.

Another important way to keep requirements achievable using COTS is to avoid imposing requirements at too low a level. Why should you impose board level requirements at all? Requirements for performance at the board level should be allocated based on operational need. For example, the operational need is that a system may need to operate anywhere in the world within 30 minutes after the application of power. Let the vendor allocate the specific temperature and humidity requirements to the boards; the ambient temperature and humidity might be controllable. Low level requirements too often prohibit the system integrator from finding other preferable solutions.

To successfully use top level requirements, the IPT needs to develop a close working relationship when deriving the lower level requirements. This is especially important for User System Interface (USI) functions. Prototyping has always proven to be an effective technique to allow users to see what the system will do and allow them to contribute to the fine-tuning of the detailed requirements. In the past, it was prudent to abandon the software developed for prototypes after the requirements were established. However, now there are COTS tools that allow the reuse of prototype Graphics User Interface (GUI) code that defines the appearance of a screen into a real-time application.

This whole concept of evaluating requirements and schedule in relation to their costs is embraced in another current DOD initiative called Cost-as-an-Independent-Variable (CAIV)[iv]. Encourage team members to suggest adjustments to requirements that reduce system development, operation or maintenance costs. This encouragement needs to be explicit, such as incentive fee awards for the developer or spot awards for members of the buyer's team. Public recognition of these awards is also a great motivator.

Many COTS boards manufactured can meet most of the traditional requirements, but most vendors will not go to the expense to formally qualify the boards to demonstrate that they do.

To continue with the 50 vs. 55 degree example, one possible impact might be that the environment would need some conditioning (cooling) before operation for a very small percentage of days in the hottest places in the world. The buyer and systems integrator need to decide whether avoiding a short delay for cooling before the system is operational is worth the additional cost and time to develop a custom board. Since many board vendors are very conservative in the testing and rating of their boards, another option is for the team to consider testing the boards at the higher temperature to determine whether it will operate at the required temperature.

Lessons Learned:

1. **Define requirements at as high a level as possible. Try to keep requirements operationally oriented.**

2. **Work with the vendor to identify cost driving requirements. Do not be afraid to reconsider requirements that force a custom board and adjust those requirements for an affordable solution.**

3. **Use placeholder requirements to conduct fair and open competition for those items that may be inadvertently large cost drivers.**

2.2 COTS System Design

The design methodology for COTS is different than for custom developed components.

The requirements for traditional MIL-SPEC design and its reviews evolved over many years for the review of custom built hardware and software. Today's acquisition initiatives promote a much more flexible approach to systems development. COTS system developments need to take advantage of that flexibility. Systems that use COTS require some modification to the traditional processes. The general approach for Preliminary Design should be "What is the top level architecture and what are its candidate parts" and for Detailed Design: "What parts should we buy?"

In general this makes the design process leading up to a Preliminary Design Review (PDR) an industry survey process which may also identify CAIV issues (as previously discussed). The preliminary design process may revert to a more traditional approach, if it is determined that there are no acceptable COTS products to meet key operational requirements.

This proposed philosophy necessitates a rework to the traditional design time line for systems using COTS items. In general, some of the activities needed to answer those questions were traditionally done later in the process.

Stressing system requirements should be identified. The top-level system architecture should isolate the cause of the stress from the remained of the system. For example, a device that requires frequent interrupts or very time acknowledgement of interrupts can be serviced by a separate microprocessor to protect the remaining system functions from the demands of a single device.

By forming teams consisting of hardware engineers, systems engineers and software engineers, requirements can be allocated without creating insurmountable hurdles for other activities. The allocation of requirements to the components can be an iterative one. As the evaluation proceeds, reallocation may be prudent if it defines a wider range of components. The entire system should be modeled to validate the top-level architecture. This process may identify cost driving requirements that should be worked within the framework established for that project.

Any placeholder requirements should be defined by the completion of the preliminary design phase, unless there is a very specific reason to continue a tradeoff study during the detailed design.

Lessons Learned:

1. **Multi-disciplinary teams and modeling are important tools in allocating requirements.**

2.2.1 Software Preliminary Design

The more requirements that COTS software can fulfill, the less custom software will need to be developed. COTS software makes sense if it substantially reduces the amount of custom software that needs to be developed. Without these potential savings, it is better to develop software and maintain control of the source and data rights.[v]

More than technical factors need to be considered when selecting COTS software candidates. The financial stability of the vendor, the vendors ability and history of solving user problems and their willingness to establish a long term support arrangement are also key qualifying factors. This long-term support is necessary to correct problems that are discovered later in the system's life cycle. Some developers have made arrangements to obtain the data rights for each product's source code, if a product's developer abandons the product or goes bankrupt.

For COTS software packages, generally there is not enough data available from the vendor in sufficient detail to determine whether the product can meet the basic need. The vendor data can be augmented with data collected from current users; but given the frequency of change of many software products, most user data available will be for older releases and will be of limited value. This requires the system integrator to obtain single copies of each product to be considered. Using the actual software, an evaluation can be conducted to determine how applicable the package is to the requirements. This initial evaluation should also consider gross compatibility issues with other software packages being considered for other requirements and some code generated from the software development environment (at least the compiler) that will be used. If the software development environment is also being selected, that selection process increases the combinations of possibilities to be evaluated.

One scenario for this phase could be a project that wants to try to select a COTS operating system, a COTS data manager, and a COTS Graphics User Interface (GUI). The range of candidate COTS operating systems could be narrowed by the real-time response requirements of the system. Then only those data managers and GUIs compatible with at least one of the remaining candidate operating systems need be considered further. The remaining candidates can be incorporated into strawman systems to evaluate how well these packages meet the requirement and whether they can work together.

Obviously, this phase can be streamlined if the system integrator can identify bundles of COTS software that already have been integrated by other projects. Any previously integrated bundles can be tested against project requirements. Less time can be spent evaluating the compatibility of those bundles. Even if a bundle has been used successfully for many years in another project, the compatibility issue is not totally resolved. Hidden compatibility issues or disqualifying bugs may still exist if the features your project needs are not the same features previously used.

Clearly, the quicker the range of candidates can be reduced; the fewer combinations need to be considered. While this task imposes a large effort during the preliminary design phase, it completes some tasks traditionally performed much later during software component integration and test phase. This "early" software integration helps to reduce development risk by identifying integration problems early.

Award fee during this phase should be partially based on the system integrator's success at identifying candidates that will minimize development cost and risk.

Lessons Learned:

1. **Select only COTS software candidates that will result in substantial savings in the amount of software that will need to be developed.**

2. **Balance any potential savings against the risks and loss of control that will result in using COTS software.**

3. **Consider more than just technical issues when selecting candidates.**

2.2.2 Hardware Preliminary Design

The preliminary design of COTS hardware is more of a paper exercise. The process for laying out the overall size, power, and cooling is fed from data collected from the individual card candidates balanced against the overall physical limitations. Developing the list of candidates requires surveying the market to identify potential candidate items and requesting detailed information on those items. The process of sizing the backplane, memory, storage, and processor speed does not vary much from custom design; however, sometimes the allocation of requirements between components may be affected by what is available on the market.

When conducting market surveys, there will be gaps in the data available to form a candidate list. Many times this initial survey activity will be frustrating since the compliance matrix will identify more questions than definitive answers. Even if a manufacturer claims compliance, the product may not meet the requirement. Marketing claims are often only true from a limited viewpoint. For example, a board may claim

100% fault detection capability, but detection data may reside only in internal registers that may not be accessible by application software. Another possibility is that the 100% claim may only include the board itself and not its interface to the chassis.

Another key point is that Non Development Item (NDI) hardware is not as mature as COTS. These items have been developed by vendors in an attempt to find a market for the item. Several programs have treated NDI similarly to COTS with poor results. NDI products have proven that too often only a single item was able to be hand built by engineers to perform a subset of requirements in an extremely controlled environment. Usually, NDI has no logistics support, has not yet considered manufacturing issues, and needs further development. If you become the only user of an NDI product, you will also need to assume all of its life-cycle support costs. The transition of NDI hardware for use is similar to the effort required to go from a successful demonstration and validation program to a full engineering development. NDI should only be considered as a strawman design approach when custom development is required.

The lack of suitable COTS candidates is not immediately a reason to go to custom development. The rate that COTS hardware is evolving is remarkable and the lack of candidates may be solved by waiting. At the rate that processor performance, storage, memory, and data transfer capabilities are increasing, the lowest risk (and cost) approach to requirement's shortfall may be to wait for the natural evolution of the required capability. This option should only be selected if the vendors are predicting the planned capability within an acceptable time frame and there is an acceptable backup approach if the planned capability does not become reality when needed.

Award fee during this phase should be partially based on the system integrator's success at identifying candidates that will meet requirements and minimize development cost and risk. It should also consider how open the architecture is and how it will affect life cycle maintenance costs.

Lessons Learned:

1. Continually follow up on data requests to try to get as much information as you need to assess a product's ability to meet the requirement. Try to get past the marketing department to the design team to get the answers that you need.

1. If a product meets almost all of your needs and has no disqualifying restrictions, you may want to carry it on the candidate list for further consideration.

2. If no products meet your needs, try to find out what planned products are available. Waiting for a planned product to mature may be preferable to embarking on a custom development.

3. NDI is not COTS. If you want to (or need to) include NDI hardware, you need to plan the additional activities required to turn it into a product.

2.2.3 Detailed Design

In order to complete the selection of hardware components and software packages, perform a detailed requirements evaluation of each candidate product. Configure those items that seem best suited to form a candidate architecture. Next, evaluate the compatibility of all the components in the candidate architecture. When those two steps are successfully completed, validate the real-time response of the resulting architecture. Plan on obtaining loaner hardware parts and evaluation copies of software packages. To complete these steps, plan on developing a mini hardware/software testbed to complete the following activities:

Requirements Evaluation. Data obtained when conducting the market survey is usually not accurate or complete. Plan on verifying key parts of the requirements with the actual hardware/software. One common example is products that claim to meet SCSI standards. While most products can typically handle data within the median range of SCSI standard limits, many products claiming to meet the standard will not operate properly if pushed to the edge of the standards (e.g. products that are close to the SCSI line length limits). Plan on evaluating each product to obtain missing data needed for the product evaluations. Most vendors do provide BIT, but do not have traditional MIL-SPEC fault detection or isolation data. Plan on evaluating and testing each potential vendor's BIT capabilities as part of the parts selection process. Plan on augmenting the BIT to meet the overall requirement.

Compatibility Evaluation. Selection of individual components to meet requirements is not enough. An important part of the design process is to ensure that all of the software and hardware components do not conflict. Plan on putting those parts together and executing COTS software and hardware together to confirm that the design is sound before purchasing components. Include software compiled using the planned development toolset, BIT software, drivers and utilities to confirm that they will all work together.

Real-Time Validation. Most COTS operating systems and utilities are not primarily designed for real-time applications. They may optimize performance for relatively small data sets using cache or memory, but may substantially slow down when operational loads are encountered. Plan on conducting performance tests with realistic operational data loads as part of the design process. Be prepared to alter hardware selection to accommodate COTS software limitations. A good example of this issue is that some versions of UNIX may provide an acceptable real-time operating system until Virtual Memory paging begins. By sizing memory large enough to keep all programs memory resident, the performance of UNIX is greatly enhanced to meet the real-time response requirements. Data collected on this testbed should be used to update system architecture models to project performance of the full system.

Life Cycle Costs. - Balance life cycle cost issues with technical concerns. Unit Production Costs, sparing costs, software licenses costs may offer different solutions to minimize cost for a single program phase (development, production, or over the system's life cycle). The system integrator should be required to show how his proposed approach affects each phase of system costs. It should consider the parts reliability, proposed maintenance plan and include both the spare and labor costs associated with repair. For example, parts that have a higher MTBF may be a better value than those that may fail more frequently, especially if those parts are time consuming to remove and replace.

Award fee for this period should be based on an evaluation of the costs and risks associated with the system integrator's proposed design solutions.

Lessons Learned:

1. **Confirm the compatibility of selected components.**
2. **Confirm that the candidate items meet the requirements. Do not rely on vendor claims if it is possible to confirm them.**
3. **There may not be an exact fit. In some cases programs may want to conduct a Cost as an Independent Variable (CAIV) analysis to consider using the best available COTS product versus the development of a custom board.**
4. **Most vendors do not have traditional MIL-SPEC reliability data. Plan to customize the reliability program for each vendor's inputs.**

2.3 Modifying what Should be Expected at Design Reviews

The criteria for design reviews must be modified to reflect the changes introduced into the design methodology.

Professionals in the acquisition of systems must evaluate whether the proposed design will meet the requirements, as well as, the technical, cost, and schedule risk associated with the selected approach. In the past, there were two key contractually defined formal design evaluations, PDR and Critical Design Review (CDR). The criteria for these reviews were defined in MIL-STD 1521, which has been canceled as part of the acquisition reform process. While the formal definition of these activities has been eliminated, these reviews are still important milestones in the top-level DOD management review process.

Especially for systems incorporating COTS hardware and software, it is important to plan what needs to be accomplished by each review. These reviews need to be tailored to the specifics of each program. The expectation of what needs to be accomplished at each review needs to be defined early in the development cycle. If these expectations are not clearly defined in detail, the data available may not support the mandated management evaluation or may result in a less than satisfactory review result. This is another item that should be arranged between buyer and system integrator as part of the IPT process. Each IPT should define the entry criteria, presentation topics, and exit criteria for each review. Since some parts of the systems may not be COTS, different topics may be required for the non-COTS items. Incremental reviews are a good way to manage the differences between COTS and non-COTS items. After the completion of all of the increments, a capstone review is an excellent approach to summarize the result of all of the increments and to show how the increments come together to meet the overall need, and to evaluate risks at the system level.

When establishing criteria for the review of COTS items, it becomes obvious very quickly that many of the traditional MIL-STD 1521 review items are not applicable. For hardware, a key item on the CDR review was a measurement of how many drawings were completed. For software, one of the key items was the completion of detailed design documentation. Reviews of drawings and software detailed design documentation were meaningful when evaluating custom developments, but do not add to an evaluation of a COTS design activity.

The focus of the criteria for these reviews for COTS items should be tailored to answer the guidelines for the design previously recommended; Preliminary Design - "What are the candidate parts" and for Detailed Design: "What parts should we buy?"

These reviews should summarize the design and requirements tradeoff issues performed to answer the question relevant to the applicable design phase. The review should present operational performance design issues. It should also present an evaluation of how the candidate design will affect production and life cycle cost issues. Traditional detailed review items should be included to the extent that they support the spirit of the design phase guideline.

Some desired data may not be available from the COTS vendors, as previously discussed. This should not delay the design review. Instead, the system integrator should identify what key data is missing, the approach to collecting that data, and the backup approach if the data, when collected, identifies an issue.

Lessons Learned:

1. **Tailor design review requirements to the specifics of each program and the guidelines for each design phase.**

2. **Clearly define the expectations for the design reviews.**

3. **Don't be afraid to proceed if all the desired data is not available. Define a plan to obtain any missing data and contingency arrangements if the data identifies an issue.**

2.4 After the Critical Design Review

After all of the COTS capabilities and limitations are well understood, the remaining aspects of the system's development can proceed on an accelerated pace with lower risk.

The major benefits for developments using COTS occurs after the completion of detailed design. At this point, a testbed of the key hardware and software items has already been assembled and most compatibility issues have been identified. Custom application development can proceed using the established baseline and the lessons learned from the detailed design testbed. There will be new issues identified as the development proceeds, but the number of integration issues will be substantially smaller than traditional custom hardware and software developments.

Another key benefit resulting from this approach is the ability to start environmental qualification testing much earlier. Moving this testing earlier in the schedule allows more options in dealing with any problems discovered.

Modifying COTS boards should not be considered as one of those options. It will void the part's warranty and will create the requirement for an ongoing production facility. Some system integrators who have traditionally been in the hardware development business are uneasy with the pure unmodified COTS approach and may design modifications (such as a unique interface or firmware) to help ensure their continued existence. Buyers should word specifications accordingly to avoid this problem and should be prepared to challenge any non-standard modifications.

Instead of modifying boards, there are often more attractive options. These include selecting another board, asking the board's vendor for the necessary modifications, or building custom enclosures to address environmental concerns. Proper enclosure design can mitigate many vibration, shock, and temperature issues. The resulting "custom" item does not create a difficult life cycle management problem.

The other post design activity affected by the use of COTS is the approach to software testing. In the past, many projects used a combination of black box and glass box testing. When using glass box, the tester takes advantage of his knowledge of the software coding. Black box testing is designed and conducted without any knowledge of the code. Traditionally, glass box testing was used by the software coder to test each piece. After all the pieces were successfully integrated, major builds were turned over to functional test teams who used black box testing to verify requirements. The lack of insight into the design and coding of any incorporated COTS software prohibits developers from using a glass box approach during software integration[vi].

Since the ultimate goal is to provide the end user with needed capabilities at an affordable price, the program manager should keep his team focused on the end result. Use a combination of a large final Award Fee/Incentive Fee increment to provide an added motivation for timely delivery of a quality system.

Lessons Learned:

1. **Avoid modifying COTS hardware (other than non-intrusive additions such as board stiffeners or added software).**
2. **Plan to environmentally qualify at least some boards that have not tested to the complete range of the operational requirements. Consider whether testing a single item or whether a complete screening of each part is required.**
3. **Plan on dealing with the shortfalls of individual cards. Some of the items that may be required include custom enclosures to isolate vibration, additional fans, or heat sinks to help control temperature.**
4. **Plan on Black Box approach to software integration testing instead of Glass Box testing.**

3.0 Methods to Help Control the Life Cycle Cost Impact

Controlling life cycle costs remains the largest challenge for systems using COTS.

COTS obsolescence is the most significant challenge to the successful deployment and operation of any real-time system. Since no program is large enough to ensure the continued production of a COTS item, each item's obsolescence must be assumed. The COTS vendors are driven by the substantially larger commercial market. As the vendors focus on current market capabilities, spares availability and product support will diminish, resulting in less control of hardware obsolescence. This is already a major issue for the programs that are using COTS.

3.1 Issues that Drive Life Cycle Cost

Minimizing the impact of future changes is a very effective method for reducing life cycle costs.

Both software and hardware issues can become major life cycle cost drivers. To help manage these issues, it is very important to know when changes to the product line will occur and what impact the changes may have. Develop an ongoing relationship with each COTS vendor to keep abreast of changes in the product line or its support.

One of the most important COTS software products to any real-time system is the operating system. It can be very expensive to upgrade. The impact of any operating system changes must be minimized by using a highly layered architecture. Prohibit direct application interface to the operating system. If the layer interfacing to the operating system is properly designed, it can be modified to update all the interfaces to a new operating system. The use of layered architectures has always been an option, but until recently this option was not viable since it imposed a performance penalty. Today's much faster processors and substantially larger memories now allow this option to be exploited.

When using other COTS software, use the layered design philosophy to insulate that software from direct application calls[vii]. If possible, try to find a way not to allow the COTS packages to interface directly with the operating systems. If the operating system needs to be replaced, it may also force an update or change to these COTS packages. Changes to the operating system will require a new evaluation of any COTS software previously incorporated into earlier versions of the systems. Changing the operating systems also imposes major cost and performance risks.

An example that illustrates this point comes from one early real-time embedded systems COTS development. UNIX operating on a SPARC 1 was selected as the operating environment. Performance requirements were demanding and the applications were optimized to maximize throughput. This system also utilized a data management package that was tightly coupled with the operating system. As the development proceeded, it became apparent that a faster computer was desired. Fortunately, the state of the art had progressed and faster processors were available and compatible with the rest of the hardware architecture. Unfortunately, the newer faster processors were not compatible with that version of UNIX. Upgrading to a new version of the operating system was expensive since its interface to application programs had changed. To upgrade, all the operating system interfaces needed to be identified and changed, the data management package needed to be replaced, and all the calls to the data management package also needed to be identified and replaced.

Operating system upgrades may have other unplanned side effects, including forcing compiler changes or driver changes for other system boards. Any upgrade to the operating system will also require a major regression test effort. Since the operating system or compiler changes are integral to the performance of a real-time system, an extended regression testing effort will be required to re-verify the system's functionality.[viii] Safety and security critical items will need to be reexamined.

The other major life cycle cost issue is the availability of replacement boards. As boards need to be replaced, the original parts may no longer be available. Sometimes it is even difficult to determine whether the boards have changed, because many commercial vendors will change their boards without changing part numbers if the board's functionality did not change. With an open systems design, it was envisioned that the new boards would simply replace the older ones; however, this cannot be guaranteed and replacement boards needed to be re-qualified (environmental testing) and operationally tested in the target environment.

Some replacement boards come with new software drivers that are not compatible with older versions of the operating system. These drivers may or may not be compatible with the operating system version already in use. Unless a compatible driver is available, replacing a board could force an operating system upgrade. Even if functional equivalent replacement boards are available, it may be very expensive to environmentally re-qualify boards and operationally test systems as part of a normal maintenance cycle.

Some program managers have envisioned that the commercial board vendors would make provisions for the long-term support of their systems. Unfortunately, this has not happened. Instead, COTS vendors typically offer each product for a relatively short period of time (1-2 years). Traditionally, replacement parts for each product are only available until the supply is exhausted[ix].

Lessons Learned:

1. **Try to select operating systems vendors with a history of backward compatibility. If the vendor continues that trend, the operating system upgrade should be less expensive.**

2. **Design systems to minimize the changes required due to hardware or operating system changes.**

3. **COTS compilers, software tools, and any COTS embedded software may or may not be compatible with systems upgrades. Utilize these items in a fashion that will minimize the impact of future changes.**

4. **When buying parts, require that every item is identical. Have QA verify that all parts meet that requirement.**

5. **Establish very close relationships with your COTS suppliers.**

3.2 Approaches to Manage Life Cycle Costs

Part of any COTS system development must be defining and planning an approach to life cycle maintenance.

Replacing each part of a system when spares are no longer available is usually not a sound approach to life cycle maintenance. There are two other approaches to better manage this problem.

The first approach is to buy enough spares for the planned life of the system. This approach can require a large initial investment in parts, but could be the lowest risk and lowest overall life cycle cost to the program. For programs that will field multiple copies of a system, this option ensures a single system configuration and reduces configuration management issues. There are a number of scenarios that make this stockpiling an unattractive option:

1. If program requirements later change and a basic system upgrade is required, then the initially purchased spares will probably be scrapped.
2. If the spares requirements are underestimated or if the planned system life is extended, there may not be enough spares to sustain operations.
3. If the number of units is large, available near term budgets may not support a large initial buy.
4. There may be shelf life issues.

The second approach consists of regularly planned systems upgrades (Planned Product Improvements (P3I)) throughout the systems life cycle. Only enough spares are stockpiled to sustain the system until the planned upgrade is complete, thus minimizing the up front cost while maintaining an acceptable risk. Spares for the out years can be budgeted to coincide with the P3I updates. This approach is recommended for most programs.

Evolving systems requirements can also be introduced at the planned upgrade. Addressing obsolescence issues simultaneously with new requirements enables the design to confirm sufficient resources to support the new requirements while renewing the availability of spare parts. This approach allows the often substantial cost of system testing to be shared between systems evolution and maintenance budgets.

Configuration management for the transition of fielded systems can be handled like any traditional obsolescence solution.

However, the second approach also has its drawbacks. Each P3I cycle can be very expensive. It may be necessary to replace all components – even if some of the components remain functional, they may not be compatible with the newer components. For example, many of us remember buying 486 computers with VESA bus display controllers. When it was desirable to upgrade to Pentium processors, the VESA adapter (and other cards) also needed to be replaced.

Either of the options can be modified to further reduce initial cost by requiring suppliers to provide notification before they discontinue a product. The program manager can then decided whether to stockpile spares, develop an additional source for the item or initiate a P3I cycle. Experience has shown that this notification will not always be provided.

Some programs have even encountered obsolescence problems in the midst of their extended development cycles. If a program's development spans more than a couple of years, many parts may be obsolete before the completion of the operational test. Since most programs cannot purchase production parts before the successful completion of operational test, it may be illegal to purchase parts for production during EDM. The first system upgrade may be the transition from EDM to production!

Lessons Learned:

1. **Select a life cycle approach as part of the system's development.**
2. **Plan for Planned Product Improvement (P3I) programs at regular intervals to solve any compatibility problems created by forced operating system (or other) upgrades.**

4.0 Leveraging Off of Other Programs

The cost savings for COTS systems will be even greater when the real-time community takes a few more steps to take advantage of the common needs between systems.

Since the major reason for using COTS is to reduce cost, a few more methods for helping to control the cost of COTS developments are recommended. These cannot be accomplished by individual projects; rather they are concepts that will work when the entire community participates.

4.1 Sharing Data

Since the key to COTS system design is the evaluation of products, sharing data learned about the candidates with other programs will streamline design efforts.

The first of these cost savings concepts is the collection and sharing of detailed requirements data collected during the design phase. In that phase, the relevant characteristics of hardware components are evaluated using the actual hardware. COTS software is evaluated for requirements compliance and compatibility. The real cost savings are achieved when the data collected by these processes is captured and shared. As the systems development progresses, additional lessons are learned about the COTS and that information should also be captured and shared.

Many companies already capture a portion of this data. Most commonly captured are some of the "non-technical" evaluation factors about the vendors. These supplier databases are usually collected by the quality organizations and are used to identify "preferred" suppliers.

This concept should be expanded to include all technical evaluation information. If one program has gone to the time and expense of evaluating (or testing) any characteristic of a component, that information should be shared with others considering using that component. This is even more important for information learned about product incompatibilities and successful workarounds.

Many companies have not formally captured this type of information in the past because there was an informal technical network that effectively shared this kind of data. However, today with more and more projects using COTS and with the large number of mergers and reorganizations, these informal networks are losing their effectiveness.

This information, if collected, could be even more effective, if it could be shared outside of the organization collecting the data. DOD could facilitate this sharing. If each contract requires this information as a deliverable, it could be inexpensively posted to a public database accessible from the web. Users could benefit from a service that would automatically provide additional information when it becomes available for selected items.

Recently, I came across an example of how shared information could help after the design phase. A friend of mine told me about problems that he discovered when mating 5 row connectors to a VME 64 backplane. When I asked another colleague who was also using VME 64, I discovered that he had also invested time identifying and solving the same problem. This duplicate effort could have been avoided if a centralized information facility existed.

4.2 Centers of Expertise

Funding common technology evaluation centers to monitor the evolution of key technologies will provide information more quickly and less expensively than having each program study those items individually.

There is another incentive that can result in large DOD wide savings. I recommend that DOD fund centers of expertise to perform ongoing technical evaluations for key evolving COTS technologies of interest to many programs. These proposed activities could be successful if they work with interested programs to evaluate a particular technology's potential capability to meet specific program requirements.

A good example of this kind of activity occurred recently. The Naval Air Warfare Center performed a study of COTS 19" flat panel displays[x]. Flat panel displays are attractive to many aircraft programs because those displays weigh a lot less, require less power and cooling, and require less space than current displays. This study evaluated the functional suitability of the available displays to perform several different applications. While this study was not centrally funded, and had a smaller scope, it did achieve many of the benefits of the proposed activities.

Similar studies would be helpful to many programs especially if the scope is expanded to include environmental and compatibility evaluations. These studies would be most helpful if they are funded as ongoing activities, so that the evaluation data remains current. New and upgraded products should be evaluated when available[xi]. The core group performing these studies would be a very cost-effective avenue to evaluate a range of products for any unique requirements.

These types of studies should not be blindly used to mandate a single common item that all programs must use. Each program must consider the logistics as part of its component selection criteria. Using items already in the supply system does decrease cost and should be considered as a CAIV tradeoff issue.

4.3 A New Approach to Using Commonality to Control Life Cycle Costs

DOD has always recognized the savings of common subsystems, but many common programs have not been overwhelming successes. To overcome these past problems, more control of common systems must be retained by the applications.

Another major way that programs can leverage off of other programs is by developing common basic architectures. The more commonality that two programs have, the more potential savings from common spares, shared development costs, and the potential for common obsolescence solutions.

There are well known reasons why common systems are not attractive to individual program managers:

1. Technical/Requirements - Common systems were force fit into applications where they did not provide a sound engineering approach and /or did not meet key operational requirements.
2. Schedule - The schedule for the development of common systems considered the user base, but frequently provided additional constraints for the users to incorporate into their program plans.
3. Funding - The budgets for the activities contributing to the development of the common system were subject to change. When the budget for one participant changed, it substantially changed the cost effectiveness for the other user(s).

All of these issues can be summed up to the primary aversion that program managers have towards common systems - Lack of Control. Any new approach to commonality must solve this control issue. It must incentivise the participants to find a way to make it work in a cost-effective manner.

One approach is to "encourage" each program manager to negotiate commonality with other programs. This encouragement could be as simple as only partially funding each program's requirements. Each program would then need to find creative ways to accomplish their needs. After arranging a potential partnership, a Memorandum of Agreement should be established between all parties defining what each party will provide to their joint effort.

Any partnering arrangement should be contingent on a successful design review approved by all parties. This design review needs to address a sound engineering approach showing how the operational and support needs of all parties are addressed. This design review should demonstrate also how the planned development would be integrated into the schedule of all applications.

To be successful, the commonality approach must include technical support from the system integrators of each application. To eliminate conflicts and keep competing organizations focused, use an independent government agent to direct the design activity and coordinate the tasking of each application's system integrator.

DOD should protect innovative program managers who are able to design cost effective common systems. Allocating independent funding for each common system will protect the participating program from any changes in budget allocations for the other partners in the common system. Each program identifies the amount of funding that it is committed to provide and then that money is reallocated to a DOD level budget line item not available for reallocation without the consent of all parties. This type of protection is necessary to provide a reasonable measure of cost control. In the past, programs have not been able to reliably project the cost of common systems, since frequently their partners have either pulled out or delayed their participation. With today's lean budgets, it is especially important to mitigate this cost risk or PMs will refuse to participate in the development of any common systems.

5.0 Ongoing Trends

It is important for the real-time community to monitor ongoing trends and promote those that will facilitate the use of COTS.

Both hardware and software trends will affect real-time systems using COTS. The trends having the largest potential impact affect software portability. Since it is unlikely that any COTS hardware will be used for the entire life of many real-time systems, the ease or difficulty of re-hosting software will contribute significantly to life cycle costs.

Without evolving into a debate over the Ada requirement, the elimination of that requirement does leave each system with the possibility that the selected high order language (HOL) may face obsolescence. This is a major issue that could drastically affect the cost to re-host custom applications. The software support activity (either government or contractor) should develop a strong relationship with the compiler vendors actively planned support for the compiler and its supporting software environment.

The C4I community is trying to establish a rating system to evaluate the "openness" of real-time software architectures. This is a good concept and should be expanded into the definition of an "open systems standard" for software. I believe that this kind of standard could be very helpful in defining an approach to allow software packages to become as interchangeable as "open system" hardware boards.

Evolving hardware standards are also a potential life cycle cost driver. Many supposed Industry standards do not stay standard for long. Many real-time embedded systems have life cycles that will exceed the life of existing standards. One key example of this issue is the SCSI standard that has evolved through many names and revisions (SCSI, SCSI 2, fast SCSI, narrow SCSI, wide SCSI, and ultra wide SCSI). In open systems, the VME standard is evolving. VME 64 is already in use and is likely to replace the older slower standard very quickly. Industry needs to consider the impact of proposed standards changes as part of its change process.

6.0 Summary

Most of the historical reasons for avoiding COTS have been overcome by advances in technology. The remaining COTS concerns can be successfully addressed by modifications to the methods by which systems are developed and managed throughout their life cycle.

The systems development and its life cycle management are not independent. Part of the development must be the selection of a life cycle management approach. Each system's design needs to be evaluated for development costs and risk and also needs to tradeoff those concerns with life cycle cost and risk issues.

Although COTS can be effectively used today, more can be done to increase its benefits. Since DOD is committed to using COTS in real-time systems wherever possible, DOD should provide some basic common support services to facilitate the use of COTS. The real-time community also should take an active role in establishing methods that will promote this policy.

About the Author:

Mr. Rosenberg is currently an Operations Director at Sabre Systems, Inc. He manages Sabre's contributions to several Advanced Technology programs. Mr. Rosenberg has over 20 years experience developing real-time computer systems and its supporting software for a wide range of DOD programs providing expertise in software engineering, Software Independent Verification and Validation, systems engineering, acquisition management and life cycle management and planning. Mr. Rosenberg can be reached at buzzr@erols.com

[i] The Commandments of COTS: Still in Search of the Promised Land, David J. Carney and Patricia A. Oberndorf, Crosstalk, The Journal of Defense Software Engineering, May 1997

[ii] Chapter 4.4 Using IPTs in Performance-Based Contracting, Writing Performance Based RFPs, Center of Acquisition Research Technology and Education (CARTE) Inc., December 1996

[iii] Alpha Acquisition Presentation, "Reinventing Negotiations" Lessons Learned from the Lamps Block II EMD Contracting Effort, Unpublished Handout

[iv] Cost As An Independent Variable (CAIV) Policy Guidance, Memorandum for Distribution, John H. Dalton, Office of the Secretary, Department of the Navy, 16 April 1998

[v] Chapter 13 Contracting for Success, Guidelines for Successful Acquisitions and Management of Software Intensive Systems: Weapons Systems, Command and Control Systems, Management Information Systems, June 1996, Volume 1, Department of the Air Force, Software Technology Support Center

[vi] A Software Development Process for COTS-Based Information System Infrastructure, Part II Lessons Learned, Greg Fox, Steven Marcom, and Karen Lantner, Crosstalk, The Journal of Defense Software Engineering, April 1998

[vii] An Architectural Approach to Building Systems from COTS Software Components, Dr. Mark Vigder and John Dean, Canadian National Research Council Report Number 40221

[viii] The Problems Imposed by Commercial-Off-the Shelf Tools in a Multi-Baseline Software Engineering Environment -- A Laboratory Manager's Perspective, Douglas Reichl and Linda Evans, Presented to the Joint Avionics and Weapons Systems Support Software and Simulation Conference, June 1997

[ix] Engineering Off The Shelf Solutions: A Life Cycle View, Doug Belaire and Mike Eagan, Naval Engineers Journal, May 1996

[x] The Liquid Crystal Display (LCD) Evaluation (A Performance Assessment by Military Users), NAWCADPAX-98-79-TM, 13 March 1998 (draft)

[xi] The Opportunities and Complexities of Applying Commercial-Off-the-Shelf Components, Lisa Brownsword, David Carney and Trica Oberndorf, Crosstalk, The Journal of Defense Software Engineering, April 1998

BUILT-IN-TEST DIAGNOSTIC ASSESSMENT

Mark Kaufman
Ed Corpuz

Support Diagnostics and Test II

1.0 ABSTRACT

Built-In-Test (BIT) is being utilized in everything from personal computers, automobiles, and earth moving equipment to complex military weapons systems. BIT is used to determine operational status and to provide information.

System performance includes BIT performance. BIT has characteristics of traditional Automatic Test Equipment (ATE), and testing system performance means assessing BIT effectiveness. Traditional system testability metrics provide some insight, but BIT's functionality goes beyond testability. BIT can be regarded as a test system within a system. BIT can be used for Condition Based Maintenance (CBM), prognostics, or to detect a mission critical fault. Regardless of its use, the basic purpose of BIT is to determine the status of a system, which is an essential element of diagnostics.

This paper will discuss types, design, applications, and examples of BIT. In addition, BIT effectiveness metrics and BIT confidence levels will be addressed.

2.0 BACKGROUND

Why use BIT? Due to changes in technology, BIT has become an important issue. Many of these changes have come about because of decade old initiatives intended to get more bang for the buck. The increasing capabilities in systems continue to drive up complexity. With increased complexity come increases in risks and test costs. BIT will help reduce risks, as well as costs. External test system development can be minimized and the associated support and maintenance costs can be reduced as well.

2.1 BIT DEFINITION

What is BIT? It can be defined as a test that is integrated with the system and cannot be removed without disabling the system.

The function of BIT is to perform diagnostics. Diagnostics is the ability to determine the status, either operational or non-operational, of a system in a timely and effective manner. Three main actions make up diagnostics: fault detection, fault isolation, and fault recovery. Fault detection is the indication that something is wrong. Fault isolation determines the fault's location to a replaceable assembly. The final action is fault recovery, which restores the system to an operational state.

2.2 BIT TYPES

There are four types of BIT, each with advantages and disadvantages. Most missiles have a short lifetime, a few minutes in most cases. After launch, there is not enough time to make corrections; the missile is expended once it is launched. Missiles are not recovered so there cannot be a Maintenance BIT function in the same sense as for an aircraft. Initiated BIT can really only be used until the time of launch. Continuous BIT cannot interfere with missile operation, but can give an indication of the health of the missile.

Power-up/Down BIT is a test that is performed when equipment is switched on or off. This type of BIT is also referred to as Built In Self Test (BIST). Personal computers (PCs) all employ Power-up BIT. As the system comes up, basic tests of the power supply, random access memory, the disk drive, and the central processing unit are performed. Power-up tests are usually not extensive; they are only meant to detect gross faults. If the basic functions of the PC are working, the operating system is loaded and then the users software. Once the PC is fully up, tests that are more extensive can be run to detect and isolate more subtle faults. This type of BIT is limited by the amount of start-up time available. Power-up BIT has the advantage that the system is not running, and conflicts with the system's operation are therefore not a concern.

Operational or **Continuous BIT** are tests that are included and integrated as part of system functions, and are either running all the time or being initiated automatically. This type of BIT must live with the system operating. BIT that tests control circuits must coexist with and function concurrently with the system's operation. Systems that are operating over long periods can benefit from this type of BIT. One example of continuous BIT is in the new six cylinder engines from Mercedes Benz. A sensor tests the engine oil and indicates when it is time to change the oil. The maximum is 18,000 miles, so a user could get a "Change Oil" indicator at a few thousand miles or much later depending on where the car was driven, how it was maintained, and the condition of the oil[1]. This is also an example of CBM (see the BIT APPLICATIONS section below).

Initiated BIT is commanded to start. An example of Initiated BIT would be a pilot pressing the "TEST" button. A well known example of Initiated BIT is the "level three diagnostic" used on the Federation Starship Enterprise from the television series "Startrek, The Next Generation." This type of BIT can be run during idle times, or prior to a critical operation. A user or controlling computer can be used to start the BIT.

Maintenance BIT is used after fault detection. This type of BIT has extensive tests and detailed fault isolation. When maintenance BIT is used, the system is in a maintenance mode and there are no conflicts with operation. In addition, the system can be reconfigured for the tests, or supplemental equipment can be used to enhance the tests.

3.0 BIT APPLICATIONS

BIT applications include CBM, BIST, and monitoring from the system level down to the integrated circuit. CBM incorporates sensors installed in systems to determine when there are changes that will affect operations or performance, and if maintenance needs to be performed. This concept does away with standardized maintenance intervals, provides information as to when is the best time to perform a maintenance action or repair, and does away with costly unnecessary actions. CBM will also give a "heads up" on critical areas of a system and provide information to avoid catastrophic failures.

The Caterpillar and White Freightliner companies use a form of CBM by installing sensors in many areas of their vehicles. Sensor data is relayed via satellite to a centralized maintenance database. The data is analyzed and then used to make maintenance decisions (i.e., replace air filter). Impending catastrophic failures are detected and in some cases, the system is shut down before the damage becomes extensive.

Another application of CBM and condition monitoring will be used on the Navy's new surface combatant, DD 21. This new ship, currently under development, will employ hundreds of thousands of sensors to gather information on operations, combat, and weapons systems. DD 21 will reduce ship personnel from today's 350 to less than 100. The sensors will be "standing watch" over equipment and the sensor data will be used to determine status, performance, and maintenance requirements.

Historically, military aircraft have utilized various forms of BIT. Today, the B1-B uses a tape recorder to capture all BIT information during a mission. The data is then analyzed on the ground after completion of the mission. Although the BIT is in continuous operation, it is used only for maintenance. The F/A-18 utilizes BIT to test the control surfaces of the aircraft prior to flight. A repair or replace decision is then made based on the results. This test is initiated by the aircraft computer as part of its pre-flight routine.

BIT has been used on various types of systems with different levels of complexity and to varying degrees. The question remains, after the requirement for BIT has been developed, how effective is it? Will it find critical faults and provide the information on how to mitigate them. You also need to know how to design your system in a way to effectively test itself without putting additional burden on reliability or cost. Balancing test effectiveness, reliability, and costs can be difficult without applying some criteria that can measure all three.

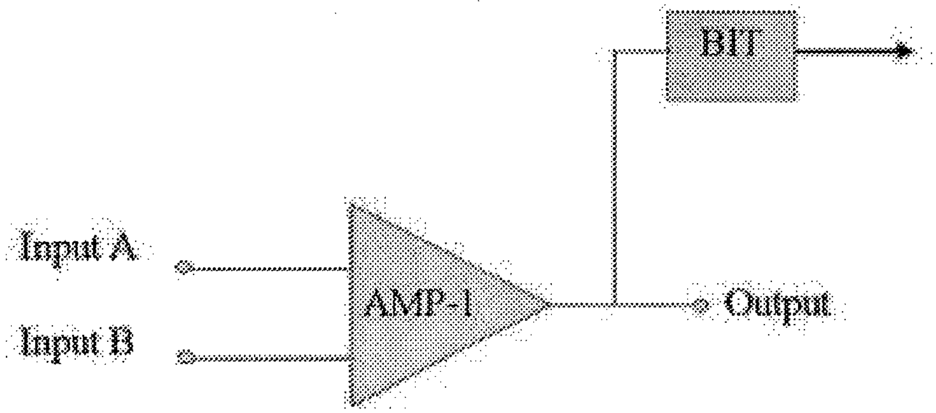


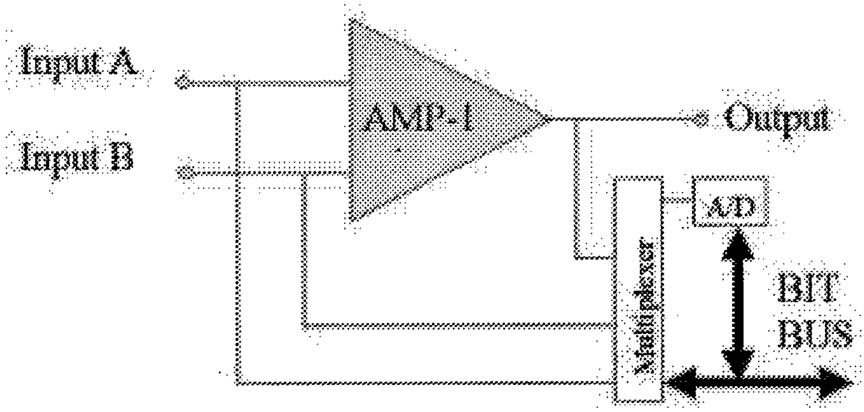
Figure 1. Simple BIT for Amplifier

3.1 BIT DESIGN EXAMPLES

These examples show several different approaches to BIT for an amplifier. The objective is to show a few approaches and their benefits. Each example determines the status of the amplifier, but with varying degrees of confidence and reliability. The circuit in Figure 1 is a two input fixed gain amplifier. A simple BIT circuit with level detection circuitry and latches is included to monitor the amplifier's output. This will give an indication when the output of the amplifier is outside a set of limits, but does not isolate the fault.

If one of the inputs to the amplifier is receiving a faulty signal, the output of the amplifier will be out of specification. There could also be transients that could cause the level detecting circuitry to give a false positive.

Figure 2 shows the next variation of the BIT design. If the output and both inputs were monitored using an A/D converter, a multiplexer, and a "BIT" computer, a BIT indication will have a higher confidence level. With both inputs known and the gain of the amplifier known, then the appropriate output voltage can be calculated. If the measurement circuitry is fast enough, transients can be recorded for later analysis. This would help



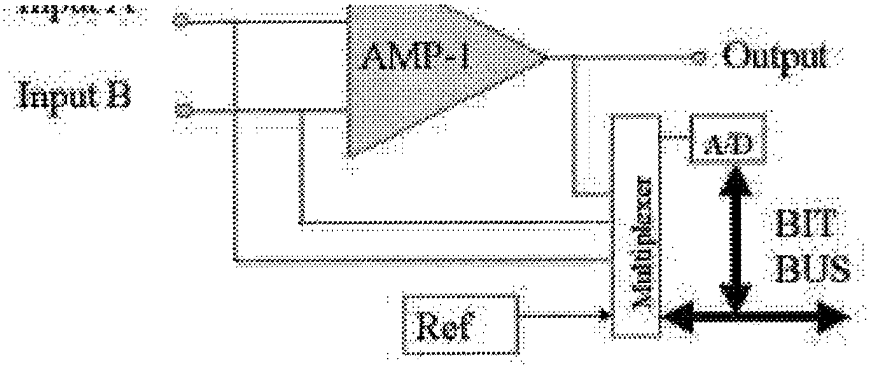


Figure 3. BIT BUS with Reference for Amplifier

isolate an intermittent failure.

The BIT design in Figure 3 adds a reference to test the operation of the BIT itself. If the reference measurement remains the same then there is a high probability the BIT is working correctly. Up to this point the BIT has only been monitoring or observing. The next version of the BIT, Figure 4, includes control circuits. If known signals are injected into the amplifier and the output is measured, the condition of the amplifier is easily determined. One cost of this approach is that it is difficult to implement for Continuous BIT without interfering with system operation. The technique of Figure 4 works best for Initiated BIT. Each of the examples have merit, and each would be an appropriate choice under the right circumstances. Each design does check the amplifier with varying degrees of completeness and complexity.

Adding the control capabilities to the BIT bus illustrates a centralized BIT architecture. A discussion of a centralized BIT architecture as part of an embedded maintenance subsystem is available in "System Test and Diagnosis."[2]

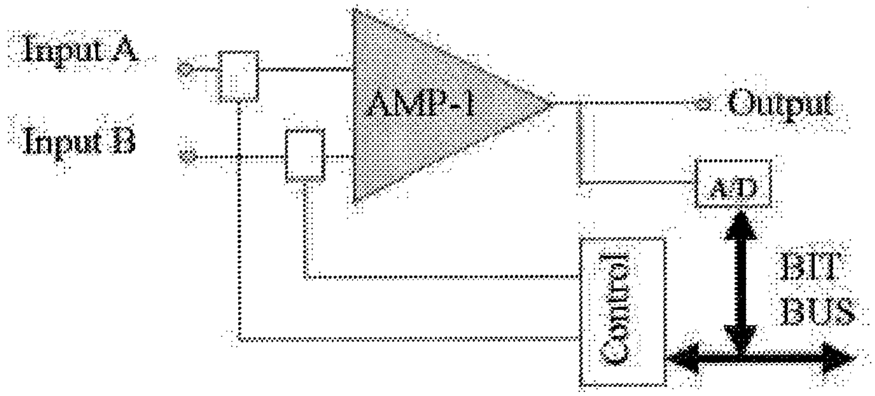


Figure 4. Input Control BIT for Amplifier

4.0 BIT EFFECTIVENESS METRICS

There are two essential elements of BIT effectiveness. BIT placement in each function is the first element. The second element is how well the test is performed. How do you know if the BIT is doing a good job? BIT effectiveness cannot be determined without knowing the characteristics of the system. These characteristics can be in the form of; system complexity, test coverage, test costs, overall system development, production, and support costs. Knowledge gained from these characteristics is very important during the design and development of the system. This knowledge can help determine the appropriate diagnostic strategy.

The following discussion of terms is based on MIL-STD-1309, Definitions of Terms for Test, Measurement & Diagnostic Equipment, and on the Institute of Electrical and Electronics Engineers (IEEE) Standard Dictionary of Electrical and Electronics[3]. These terms also reflect the work on testability and diagnostics metrics being done in the IEEE SCC-20 standards committee; Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE). A proposed standard, IEEE P1522: Testability and Diagnosability Characteristics and Metrics is of particular interest. A complete description of the effort can be obtained from the AI-ESTATE web site.[4]

BIT Coverage is the amount of the system tested by BIT. BIT coverage could be expressed as a percentage of overall system test capability, number of functions tested, or as a percentage of detectable faults or total faults.

Ambiguity Groups deals with a group of assemblies or actions. The ideal ambiguity group is one, there is only one action to be performed or assembly replaced to restore the system to operation. This metric is very important when planning for maintenance and repair actions.

Observability/Controllability/Accessibility are very important aspects of BIT. Important information can be obtained by observing a circuit. More information can be gained by being able to control the inputs to the circuit and then observing the outputs. This metrics can be expressed as a percentage of critical functions or total faults.

Feedback Loops can cause major problems in testing. A controlling feedback loop (i.e. opening them) is an essential point here. This goes back to the issues of controllability and observability discussed above. This metric can be expressed as a percentage of total feedback loops that can be broken. Although this metric does not measure BIT directly, it is an indicator of how easily faults are isolated. Nested feedback loops make fault isolation difficult.

BIT Time refers to how long does the BIT take. BIT is usually faster than off-line test equipment. BIT will run at the same speed as the system itself and will not have to involve the extended connections and access times of off-line test equipment. BIT time would give valuable information for production, maintenance, and operational planning purposes. A diagnostic strategy can be adjusted to optimize on time, cost, or other factors.

BIT Cost can refer to several things. Costs can be associated with test or BIT development, test time, set up time, cycle time, etc. There may be some conditions to test cost like having to wait until the system cools off due to the design and characteristics of the hardware. The system allows for only limited power-on time before overheating takes place. Performing BIT and not involving slower off-line test equipment will reduce cycle and overall test time. The savings from developing and performing BIT versus off-line test can be used to determine overall system lifecycle costs.

Fault Detection/Isolation/Recovery can be expressed as time measurements or probabilities. As a time metric, this would be the time it takes to detect, isolate, and recover from faults. This metric is often used in formulas to calculate the operational availability of systems (for example, mean time to repair). The probabilities of fault detection, isolation, and recovery can be calculated and utilized as test and hardware design metrics, as well as providing additional maintenance planning information.

Percentage of Total Faults Detected by BIT can be expressed as the percentage of total possible faults, the total detectable faults, or the total mission critical faults.

As seen from the metrics described above, important information can be retrieved so proper decisions can be made during the development and testing of the system. How well BIT performs can be determined early on in the development process.

5.0 BIT CONFIDENCE

As we design a system with BIT, a determination can be made on how effective BIT is. The question that remains is how much confidence do we have in our BIT and does it have a negative effect on the total reliability of the system?

A statement can be made that BIT must be much more reliable than the system. The BIT failure rate should be much less than the system failure rate. BIT determines the health of the system and must have a very small failure rate. System reliability can be calculated using traditional methods. Failure Modes and Effects Analysis (FMEA) can be conducted to determine critical functions and failure modes. Reliability of critical functions can then be established and be compared to the overall system reliability:

$$\frac{\text{Critical Function Reliability}}{\text{System Reliability}}$$

The critical function reliability takes into account the areas of the system utilized (hardware) and the portion of software. BIT can be considered a critical function, which utilizes areas of hardware and software. The time metric and probabilities of BIT functions (fault detection/isolation/recovery) can be used to calculate BIT reliability and to determine overall system availability. With this approach, analyses and trade-off studies can be conducted to determine the effects of BIT on the system, the advantage of improved availability, or the disadvantage of reduced reliability due to "burdening" the system with added BIT circuitry.

This is just one method to determine BIT confidence. Further investigations and discussions with members of the test and design community may uncover methods that are more effective. The requirement for BIT in complex systems, along with technology demands, will drive the need for increased reliability. BIT can increase reliability and availability if integrated into a system properly, by utilizing the information provided by metrics.

6.0 SUMMARY

Built-In-Test is an effective function in determining the status and condition of a system. It is used in power up conditions and various maintenance situations. BIT, if applied properly, will help reduce the risks and costs associated with complex systems. As it is necessary to assess the performance of systems and effectiveness of critical functions, it will also be important to assess BIT effectiveness. The metrics and methods discussed above, will assist in the effective design and development of new systems with BIT, and allow for the assessment of this critical function throughout the whole process.

7.0 REFERENCES

[1] "Motor Trend," December 1997," '98 Motor Trend Truck of the Year, Mercedes-Benz M-Class", page 63.

[2] "System Test and Diagnosis," William R. Simpson, John W. Sheppard, page 229.

[3] Standard Dictionary of Electrical and Electronics Terms (IEEE Std 100) 1996

[4] IEEE AI-ESTATE Web page: http://grouper.ieee.org/groups/1232/

Title: Special Operations Forces (SOF) Extendable Integration Support Environment (EISE) Integrated Product Team (IPT) Process: A Non-traditional Approach to Laboratory Development

Presenter: Clay Mims WR-ALC/LYSBD

Track:

Abstract:

The process used for the engineering development, testing, and operation of the Special Operations Forces (SOF) Extendable Integration Support Environment (EISE) is a success story unparalleled at the Warner Robins Air Logistics Center. The SOF EISE is a software engineering, test, and evaluation laboratory for maintenance and testing of Operational Flight Programs (OFPs) for avionics embedded computer systems flown aboard SOF aircraft.

To meet the operational need for an OFP test capability in two years, with a limited budget, it was clear that neither "traditionalî government/contractor relationships, nor a traditional government ìin-houseî organic approach would be successful. The program leadership determined the best approach would be to forge a non-traditional teamwork partnership with the incumbent contractor (TRW) to complete the SOF EISE, thus the SOF EISE IPT was formed.

What is the SOF EISE?

The SOF EISE is a software engineering, test, and evaluation laboratory for maintenance and testing of Operational Flight Programs (OFPs) for avionics embedded computer systems flown aboard SOF aircraft. The laboratory simultaneously simulates the MH-53J, AC-130H, MC-130H, and MC-130E. The SOF EISE provides an open and flexible architecture that allows for expansion of weapon system modifications and adding future aircraft.

Figure 1: SOF EISE LABORATORY

The SOF EISE is used by the SOF test engineers to conduct reliable and repeatable testing of avionics embedded computer systems software prior to aircraft ground and flight test. The data recording and reduction capabilities allow the test engineers and OFP maintainers to thoroughly analyze test results and to conduct software changes in a controlled environment.

Program Background

Following almost 8 years of programmatic and technical turmoil in developing the SOF EISE facility using traditional government/contractor interaction, the SOF System Program Office (WR-ALC/LU) ìgave upî on the development. The SOF SPO issued a stop-work order to their contractor (TRW) and turned the program over to the Avionics Management Directorate (WR-ALC/LY) for completion.

Different Approach Required

Reviewing the scope and troubled history of the SOF EISE, LYS leadership saw that the LYS organization had never tackled a task of this magnitude and lacked critical expertise in SOF EISE system architecture and simulation systems. Similarly, the program team saw that TRW lacked sufficient experience with aircraft OFP testing and required assistance with critical software development disciplines such as formal peer reviews and configuration management.

The program team decided to conduct a Configuration Audit/Requirements Traceability Analysis (CARTA) of the program to facilitate the transfer of development responsibility. The purpose of the CARTA was twofold:

to determine the current state of development of the system (e.g. configuration audit)

to trace the implementation of the customer (WR-ALC/LU) requirements in the system design as currently being implemented by TRW (e.g. requirements traceability analysis)

Information collected during the CARTA was used by LYS to plan the execution of the remaining development of the SOF EISE. Following the completion of these efforts, a Program Management Review (PMR)

SOF EISE IPT

To meet the operational need for an OFP test capability in two years, with a limited budget, it was clear that neither "traditionalî government/contractor relationships, nor a traditional government ìin-houseî organic approach would be successful. The program leadership determined the best approach would be to forge a non-traditional teamwork partnership with the incumbent contractor (TRW) to complete the SOF EISE. Leveraging TRWís expertise in simulation systems with LYS expertise with aircraft OFP software and Software Engineering Institute Capability Maturity Model (CMM) Level 3 software development processes and procedures the SOF EISE IPT was formed.

Figure 2: SOF EISE IPT

Team Empowerment

Obtaining empowerment to proceed with this non-traditional approach required buy-in from several stakeholders. The first stakeholders were the team members themselves. LYS personnel were uncomfortable assuming responsibility for a project of this magnitude, particularly given its reputation as a troubled program. TRW personnel were concerned over the ramifications of the LU stop work order, especially its impacts on their job security. Another stakeholder was the contracting community. Serious issues of contract scope, responsibility, and billing had to be resolved. This was accomplished using a more flexible Time and Materials ñ type contracting arrangement in place of the previous Cost Plus Fixed Fee contract. Most important, organizational leadership in both WR-ALC/LY and LU had to be persuaded that the teamwork/partnership could be successful. And lastly, the operational customer, Air Force Special Operations Command, had to agree to continue to fund a program that had already taken 8 years, and had yet to produce results.

WR-ALC/LU and LY jointly briefed the proposal to the Center Commander (WR-ALC/CC), who gave final approval to the approach, and granted the team maximum flexibility within the established program baselines to accomplish the assigned taskings. Following WR-ALC/CC approval, WR-ALC/LU provided information briefings to their warfighter customers at Headquarters, Air Force Special Operations Command, to gain concurrence and support for the effort.

Keys To Success

Key to the SOF EISE success is communication. Before IPT formation, communication between contractor and government was very poor. Requests for documents and information took months to be answered. In the SOF EISE IPT, government engineers responsible for providing information sit side by side with both TRW and LYS engineers who require it. Communication is facilitated by weekly project meetings for each major system segment in development. At least three times a week team members are face to face with both LYS and TRW program management. Jointly chaired configuration control boards and design reviews ensure key aspects of the project remain focused. Joint software peer reviews and software documentation reviews ensure the technical aspects of the project are meeting design goals. Specialized systems expertise is provided by several support contractors, who are also members of the IPT. And finally, a joint test team, with TRW team members ensuring specification compliance and LYS team members ensuring the operational suitability of the system, helps to keep the SOF EISE focused on its overall objective: to build a state-of-the-art software test and evaluation facility for the Special Operations Forces warfighter.

Prior to the IPT, government insight into the development of the SOF EISE was limited to infrequent formal reviews and occasional ìwalk throughs.î Immediately after the IPT was formed, LYS moved most of its team members into the SOF EISE, side by side with the TRW team members. Day to day operations saw a blending of the best practices from each organization. Responsibilities were determined on an individual, not organizational basis. LYS engineers sat in TRW software peer reviews, not as ìgovernment representatives,î but as true peers, able to make inputs on design and quality directly to the engineers, and not ìfilteredî by management and contracting. The primary touchstone quickly changed from ìWhat does the contract say?î to ìWhat makes the most sense?î

The daily customers of the SOF EISE development effort are the OFP test engineers. Before the IPT was formed these test engineers were allowed into the SOF EISE only on a ìnon-interference basis.î With the formation of the IPT, test engineers became full members, with key roles in the design, testing and operation of the system. The result: SOF EISE does the job they need it to do, and, with their help, it was done right the first time.

Cost, Time, and Energy Savings

At the time WR-ALC/LU issued the stop work order, they estimated that it would take approximately 12 months to award a contract to replace TRW with another company. It was further estimated that the new contractor would take 18 to 24 months to learn the existing system before they would be able to deliver a suitable product. Similarly, initial LYS estimates to perform the entire SOF EISE project in-house called for about 3 years to build the first segment, followed by two years for each additional segment. These estimates assumed that LYS could conduct a large and rapid staff-up to handle the new workload (something it has not been able to do, given the current emphasis on defense downsizing). Thus, by forging the strategic teamwork partnership, the SOF EISE IPT was able to begin work immediately (saving the 12 month contracting lead time) and was able to deliver the first segment in under 18 months, with the remaining segments following with a year, for a total time savings of over 24 months over either alternative. Also, by retaining TRW on contract, the IPT was able to constructively use $2.7M of previous year contract funding that would otherwise have been lost to the program (and would have to be replaced by WR-ALC/LU).

SOF EISE IPT Results

The results were greater than expected. The SOF EISE-Gunship Simulation Network development was seen as ìambitiousî and moderately risky by the customer at the beginning of the project. Nonetheless, by employing the teamwork/partnership perspective, the SOF EISE IPT achieved Initial Operational Capability (IOC) six weeks ahead of schedule, and on budget.

Technically, SOF EISE is a great success. OFP test engineers have deemed the SOF EISE as a ìquantum leap improvementî over the aircraft prime contractor support laboratories it is designed to replace. The system has been used to recreate flight conditions in a controlled environment and to detect software problems before flight test. The result: reduced flight test costs. In acceptance testing the facility, several latent errors were found in the fielded OFP used as the test baseline. These latent errors had previously escaped detection, despite repeated testing in the aircraft prime contractorís support laboratory.

Summary

The process by which the SOF EISE IPT was operated definitely lends itself to other organizations. Although the ìtraditionalî government/contractor relationship is the primary method of operation within the AFMC world, non-traditional government/industry teamwork partnerships are beginning to show their promise. With government and industry each applying their own unique key abilities and core competencies to the various efforts, blending and integrating these into a highly productive team requires vision that extends beyond the traditional government/contractor relationship.

Clearly, the success of the SOF EISE IPT shows that non-traditional partnerships between government and contractors can forge a synergy yielding results greater than the sum of the parts. All that is required for this synergy is: 1) a change in managerial mindset from seeing organizations as entities, instead focusing on the capabilities of individual members, 2) visionary contracting officers who see beyond the legal document and on to the results of contracted effort, and thus to use contract types more difficult to administer, but allowing more flexibility to yield greater result; and 3) senior leaders willing to empower a team, even if key aspects of the project fall to team members not under their direct authority. The SOF EISE IPT was blessed with each of these, and was thus able to deliver in less than 18 months a product that the traditional government/contractor approach was unable to produce in the previous 8 years.

TRW / WREL
- Systems Engineers
- Hardware Design & Fabrication

HQ AFSOC
- LGM (Support Plans)
- XPQ (Sys Expertise)

SOF EISE
Integrated
Product
Team

WR-ALC/LU
- Program Mgt.
- SPO Operations
- Log. & Funds Mgt

WR-ALC/LYS
- Project Management
- Software Engineers
- Software Testers

Support Contractors
- System Expertise

conducted to apprise the senior leadership of the SOF SPO and the Avionics Production Division of the SOF EISE current development status, deficiencies identified between the specified requirements and the current design, and to review the proposed development program.

Team focus was emphasized by the program leadership as a key aspect for the SOF EISE IPT to succeed. The was led to focus on the technical problems and the most optimum solutions. Technical requirements were established with the customer and the team designed the SOF EISE hardware and software to these requiremnts.

SOF EISE IPT

SPECIAL OPERATIONS FORCES EXTENDABLE INTEGRATION SUPPORT ENVIRONMENT (SOF EISE)

INTEGRATED PRODUCT TEAM

Clay Mims
WR-ALC/LYSBD
DSN 468-0227
cmims@lys.robins.af.mil

OUTLINE

- What is the SOF EISE?
- Program Background
- A Different Approach Required
- SOF EISE IPT
- Tools and Methods
- Results
- Integration Of New Processes
- Level of Leadership
- Time And Money Saved
- Cost Savings
- Exportability

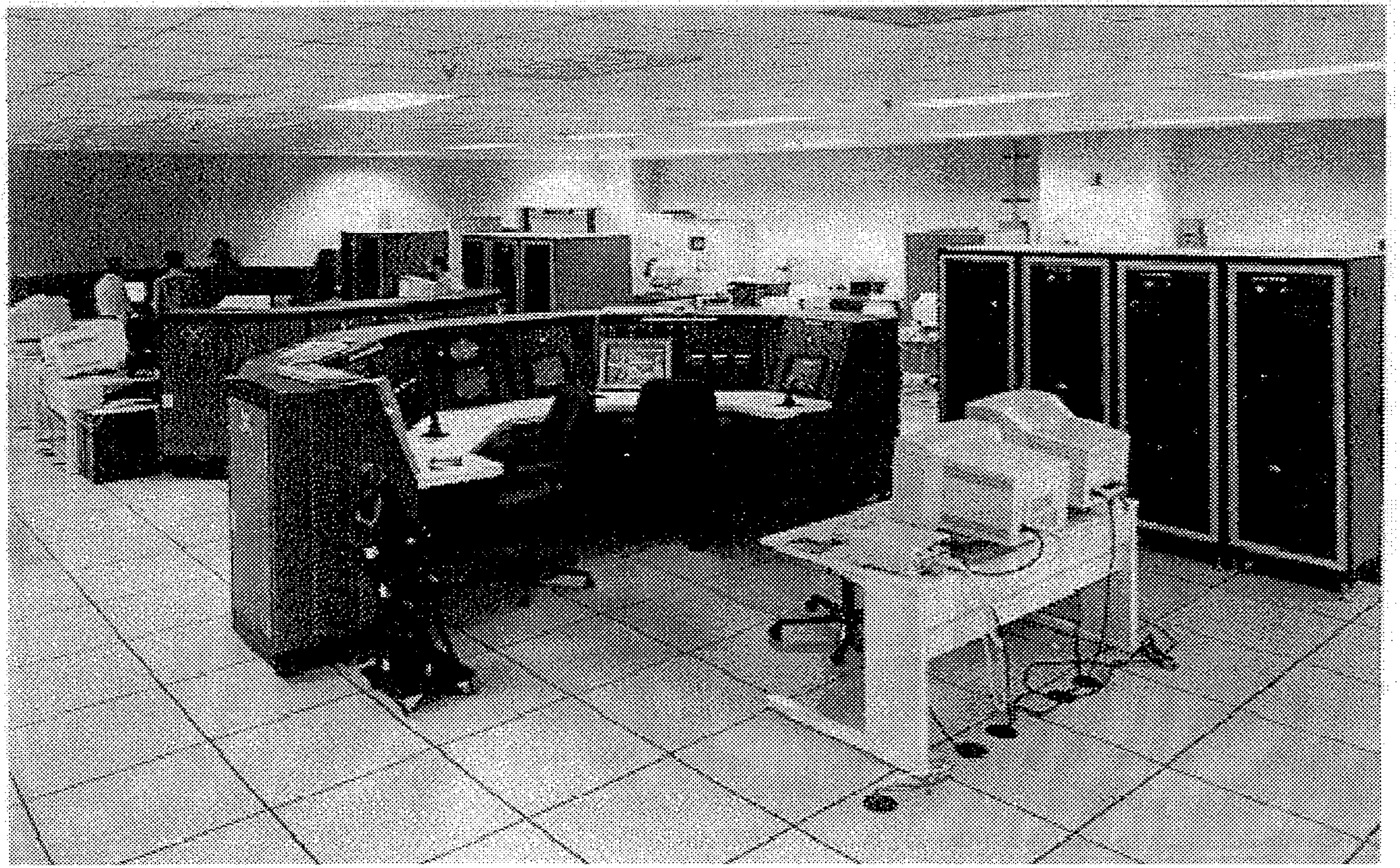
WHAT IS THE SOF EISE?

SOF EISE IPT

- State-of-the-art software development and testing laboratory at Robins AFB GA
  - Designed for development and test of Operational Flight Programs (OFP) for selected SOF aircraft
- Simultaneously simulates MH-53J, AC-130H, MC-130H and MC-130E
  - Open, flexible architecture allows expansion for weapon system modifications and adding future systems
- A reliable, repeatable test environment for OFP testing and troubleshooting

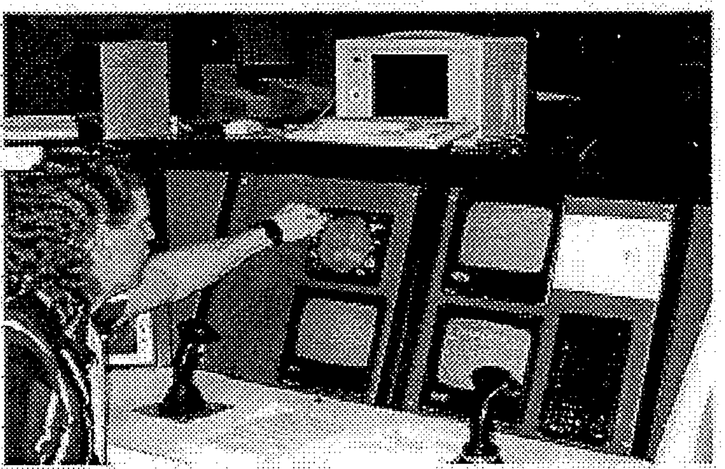
WHAT IS THE SOF EISE?

SOF EISE IPT

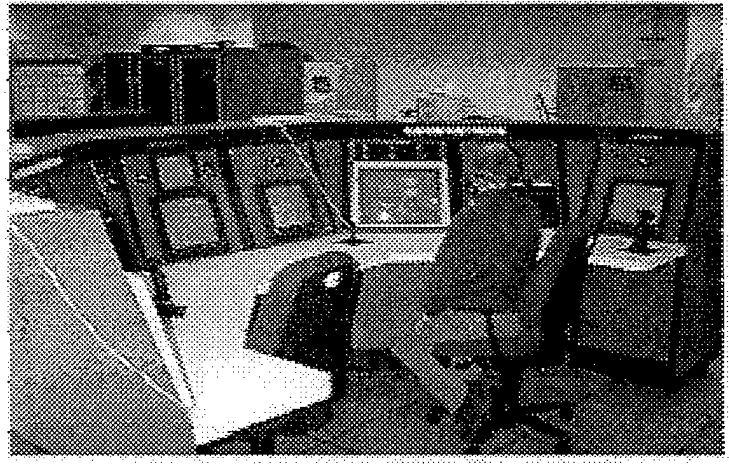


SOF EISE OFP TEST STATIONS

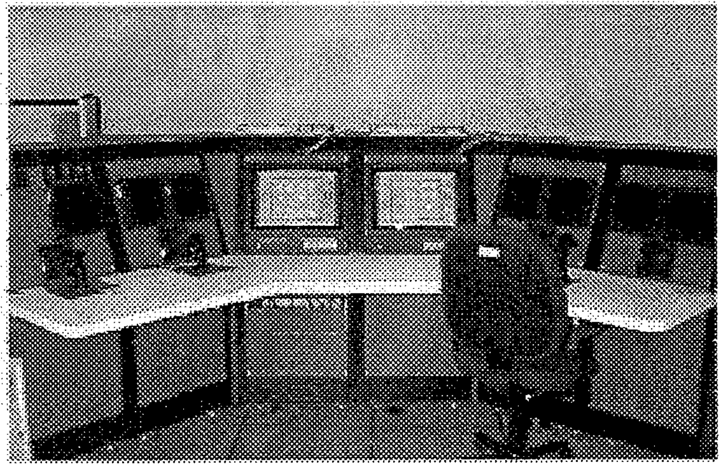
SOF EISE IPT



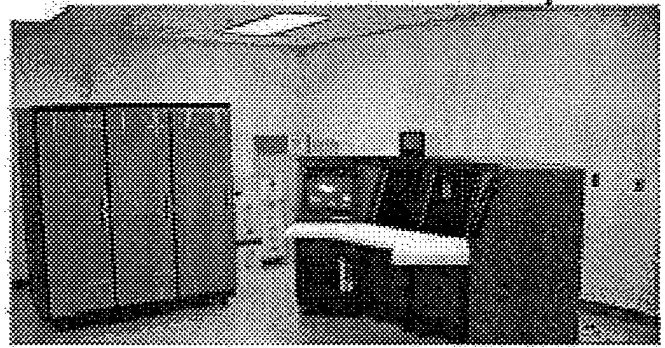
MH-53J Pave Low III



AC-130H Gunship



MC-130H Combat Talon II



MC-130E Combat Talon I

PROGRAM BACKGROUND

or "How Did We Get Into This Mess?"

- WR-ALC/LU awards development contract to TRW in 1987
  - -- Planned as a 4 year, $16 M project
- Technical and contract troubles developed early
  - -- Delays, schedule slips, funding shortages and cost overruns were numerous
- In Nov 1995, LU had lost confidence in the program
  - -- Project was in year 8, cost had grown to $39 M
  - -- One of five planned segments delivered
- LU issues "Stop Work" order, asked LY for help

A DIFFERENT APPROACH REQUIRED

or "Business As Usual Won't Work!"

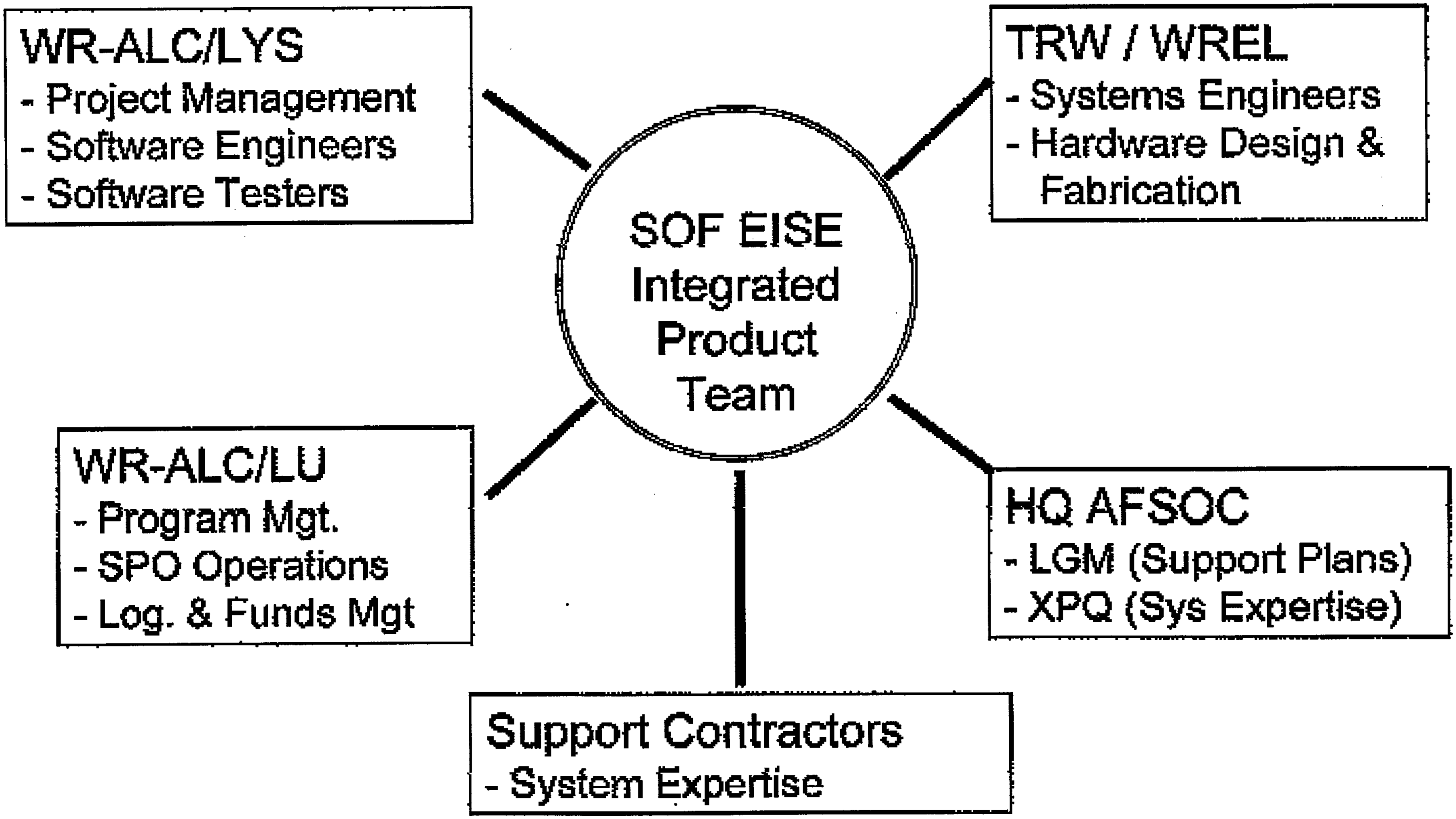
SOF EISE IPT

- WR-ALC/LY team conducts program assessment
  - -- Determines "pure organic" approach to be HIGH risk
- LY decides to forge Partnership with TRW
  - -- Based on True Integrated Product Team precepts
  - -- A "win-win" situation
- Result: A Single Team - The SOF EISE IPT
  - -- Government and contractor working on same segments
  - -- Tasks assigned on availability and qualifications, not parent organization

TEAM MEMBERSHIP

or "Who are we? - A Partnership!"

SOF EISE IPT



SOF EISE Integrated Product Team

WR-ALC/LYS
- Project Management
- Software Engineers
- Software Testers

TRW / WREL
- Systems Engineers
- Hardware Design & Fabrication

WR-ALC/LU
- Program Mgt.
- SPO Operations
- Log. & Funds Mgt

HQ AFSOC
- LGM (Support Plans)
- XPQ (Sys Expertise)

Support Contractors
- System Expertise

TOOLS AND METHODS
or "How did we make it work?"

SOF EISE IPT

Communication
Keys
To
Success
Disciplined
Processes
TEAM
Focus

RESULTS

or "Were results better or worse than expected?"

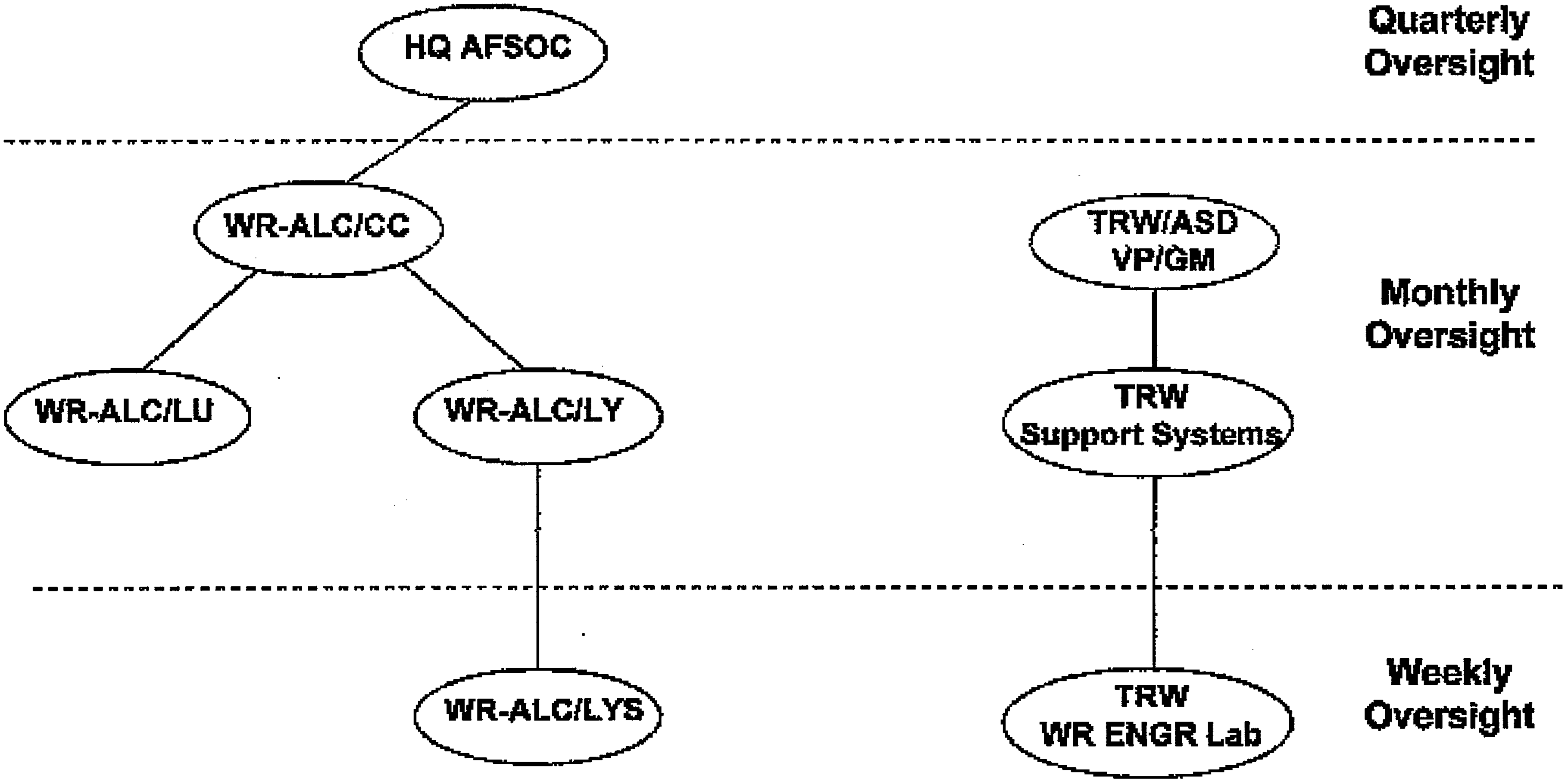
SOF EISE IPT

- Team coalesced rapidly, products flowed fast
  - -- Barriers were real, but came down rapidly
  - -- Relocation of LYS to Bldg 230 helped immensely
  - -- Team members soon gained identities from system segment, versus parent organization
- Bottom line: Gunship segment delivered ahead of schedule, on budget - in only 18 months!
- Customer Influence? -- On the team, at the table!
  - -- Before SOF EISE IPT: Test Team allowed into the lab on a "non-interference basis," (as deemed by CO)
  - -- Post IPT: Test Team at all meetings, live in lab, and are primary system operators for development and test

LEVEL OF LEADERSHIP

or "Who do we brief this week?"

SOF EISE IPT



TIME AND MONEY SAVED

or "Better, Cheaper, Faster!"

SOF EISE IPT

SOF EISE Annual Cost To Support 4 Systems
(MH-53J, AC-130H, MC-130H, MC-130E)

Manpower	$2.80 M
Hardware	$0.50 M
LRU Maintenance	$0.25 M
TOTAL	$3.55 M

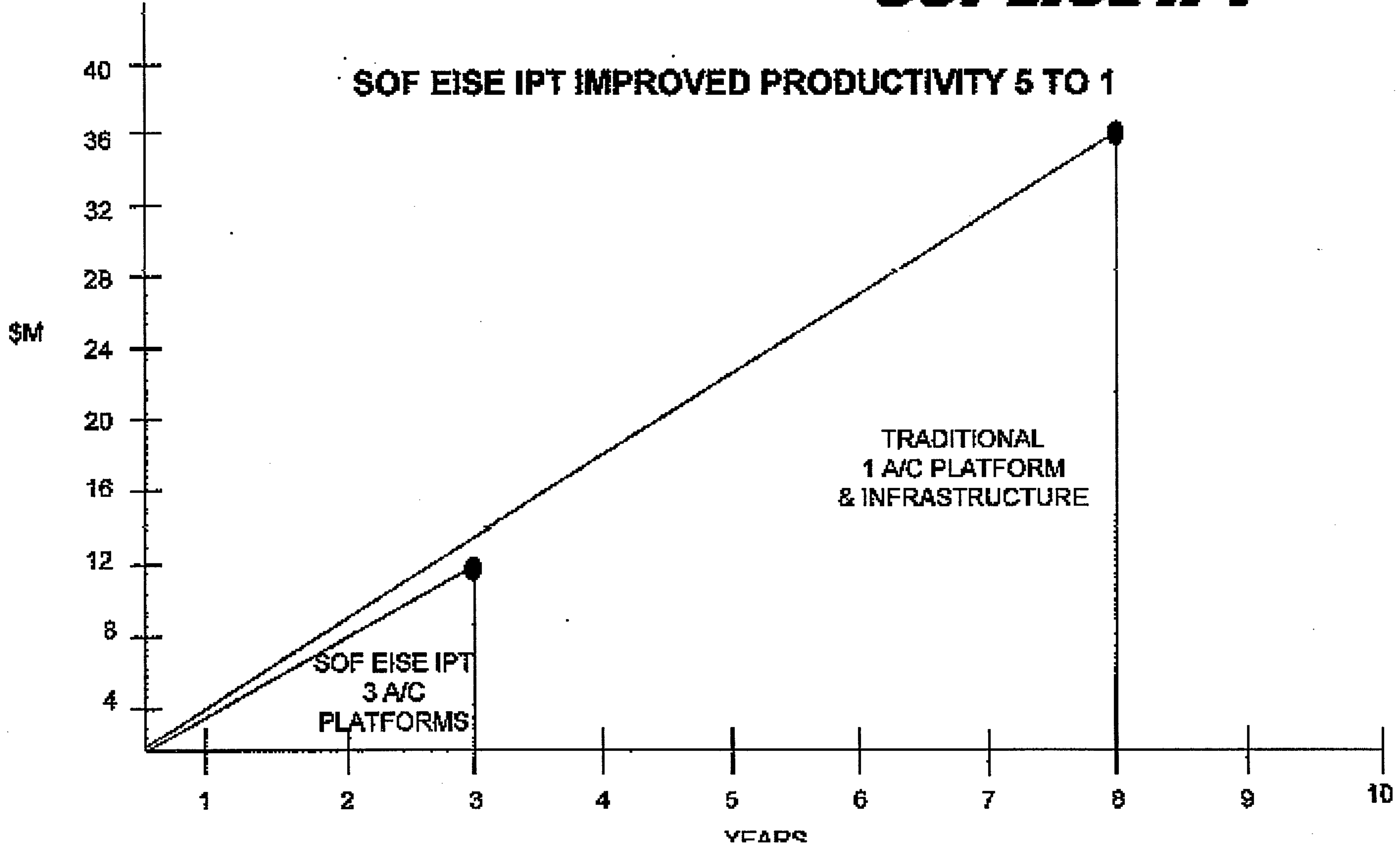
$3.55M for 4 Systems = $0.89M / Year / System

AC-130U (Traditional) Annual Software Laboratory Support Cost = $10M / Year / System

COST SAVINGS

or "SOF EISE partnership saved time and money!"

SOF EISE IPT



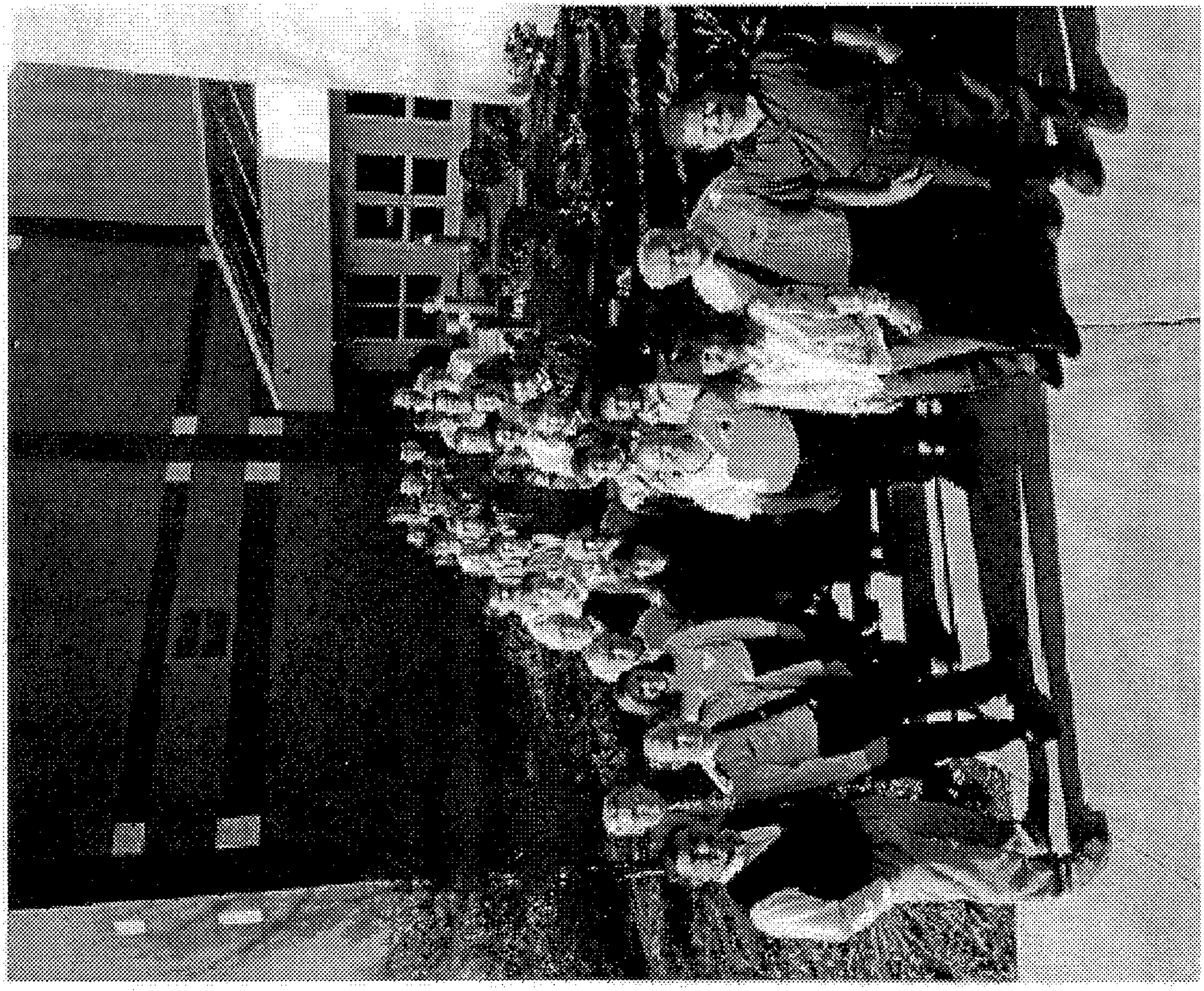
SOF EISE IPT IMPROVED PRODUCTIVITY 5 TO 1

EXPORTABILITY

or "Why doesn't everybody do it this way?"

SOF EISE IPT

- The SOF EISE IPT Partnership is an excellent model for government/industry strategic partnerships
  - -- Leverages core competencies of each member
  - -- Provides a unique synergy
  - -- Facilitates "ownership" of product very early
- How? A true paradigm shift
  - -- Change mindset from organization-based to individuals
- What's needed?
  - -- Change in managerial mindset
  - -- Visionary contracting officers
  - -- Senior leaders willing to empower a team



SOFEISEIPT

Distributed Simulation Testing for Weapons System Performance of the F/A-18 and AIM-120 AMRAAM

by
LCDR Tom Watson
Naval Weapons Test Squadron
Point Mugu, CA

I. Abstract

The Naval Air Warfare Center Weapons Division has established a long range, real-time link between the F/A-18 Weapons System Support Facility (WSSF) at China Lake, CA and the AIM-120 Hardware in the Loop (HWIL) laboratory at Point Mugu, CA. The link was established in response to a fleet demand for information on the total weapons system performance of the Hornet and AIM-120 sub-systems in an electronic jamming environment, since AIM-120 performance is very dependent on the quality of the guidance data link provided by the host aircraft. In an effort to minimize costly flight testing, the link concept was developed to obtain actual (vice simulated) aircraft radar performance and data link updates in order to more accurately assess overall performance of the aircraft/missile system.

II. Introduction

The addition of the AIM-120 Advanced Medium Range Air-to-Air Missile (AMRAAM) to the F/A-18 Hornet represented a significant increase in the warfighting capability of the aircraft. Current and evolving threats make a credible beyond-visual-range launch-and-leave missile an imperative to acceptable combat survivability in the modern air-to-air arena. Not only does the AMRAAM bring the aircraft a fast missile with good range and kinematic capability; it also brings to the table significant upgradability through software enhancements. As electronic attack threats evolve and are exploited, the missile software is enhanced to counter the new capabilities of potential adversaries. Significant simulation testing of the missile performance is done during new software development, but little testing of the total aircraft/missile systems performance together is performed by the developer.

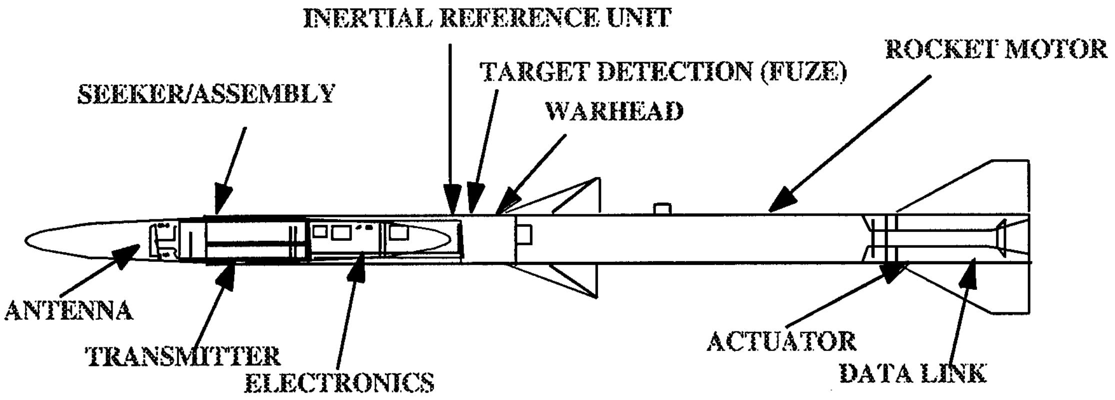
The AMRAAM design is dependent upon the quality of aircraft data link for target acquisition. To address this dependence, a series of tests to quantify and characterize the total weapons system performance in an electronic attack environment was clearly needed. This need was answered through a program that is jointly funded by the AMRAAM and F/A-18 program offices.

In recent years, there has been much discussion, planning, and experimentation in the area of distributed simulation. Under this concept, simulations of complex interactive systems that are located in separate buildings or at different geographical locations are linked electronically. One application of this concept is the connection of missile and aircraft simulations for the purpose of determining system interface issues and performance capability. Within the Naval Air Warfare Center, the AMRAAM HWIL facility at Point Mugu has been working on an electronic link with the F/A-18 Weapon System Support Facility at China Lake. The WSSF is part of the F/A-18 Weapon System Support Activity. In direct response to budget cuts and the significant cost of flight test the concept of linking the F/A-18 and AMRAAM labs was born.

III. Background

3.1 AMRAAM:

The AIM-120 is an advanced medium range missile that provides the aircraft with a beyond visual range launch-and-leave capability. It is a $6 Billion Acquisition Category 1D Joint Air Force-Navy program with the Air Force acting as the executive service. Hughes Missile Systems (now Raytheon) is the prime contractor and developer of the missile. The missile is 12 feet long and weighs 345 pounds. Figure 1 shows the basic layout of the missile. The missile is tail controlled, highly maneuverable and has excellent average velocity over the time of flight. An onboard X-band radar that operates in both high and medium pulse repetition frequencies provides terminal guidance.



PARAMETERS	AIM-120A/B	AIM-120C
LENGTH	144 IN	144 IN
DIAMETER	7 IN	7 IN
WING SPAN	21 IN	17.5 IN
CONTROL SURFACE SPAN	25 IN	17.5 IN
WEIGHT (NOM)	345 LBS	345 LBS
CG (NOM)	79.8 IN	79.8 IN

Figure 1
AMRAAM Configuration

AMRAAM has undergone significant changes since first developed and is currently being produced in the AIM-120C clipped wing version for the U.S. and the AIM-120B version for foreign sales. The clipped wing modification was required for internal carriage on the F-22 aircraft. Currently, a new rocket motor with 5 inches of extra propellant, a new warhead with more fragments and a new target detecting device are in development. Figure 2 shows the clipped wing version of the missile.

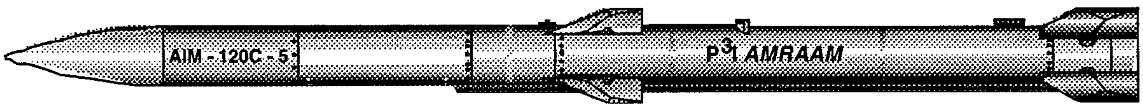


Figure 2
AIM-120C Clipped Wing

The AMRAAM operates in several modes. The preferred launch mode is the Command Inertial mode in which the missile receives targeting instructions from the aircraft through the use of an RF data link which is updated every 0.5 to 1 second depending on the launch mode of the aircraft radar. Inertial Active is a complete launch and leave mode in which the AMRAAM guides to an inertial point provided to the missile pre-launch with no updates during flight. Terminal guidance is provided by the missile's onboard radar. The missile is also capable of sorting multiple targets and picking individual targets for each missile launched against an unresolved group of targets. The missile can also be launched in a home on jam mode in a noise jamming environment and a visual mode when no radar targeting is provided to the missile. Figure 3 depicts the various launch modes:

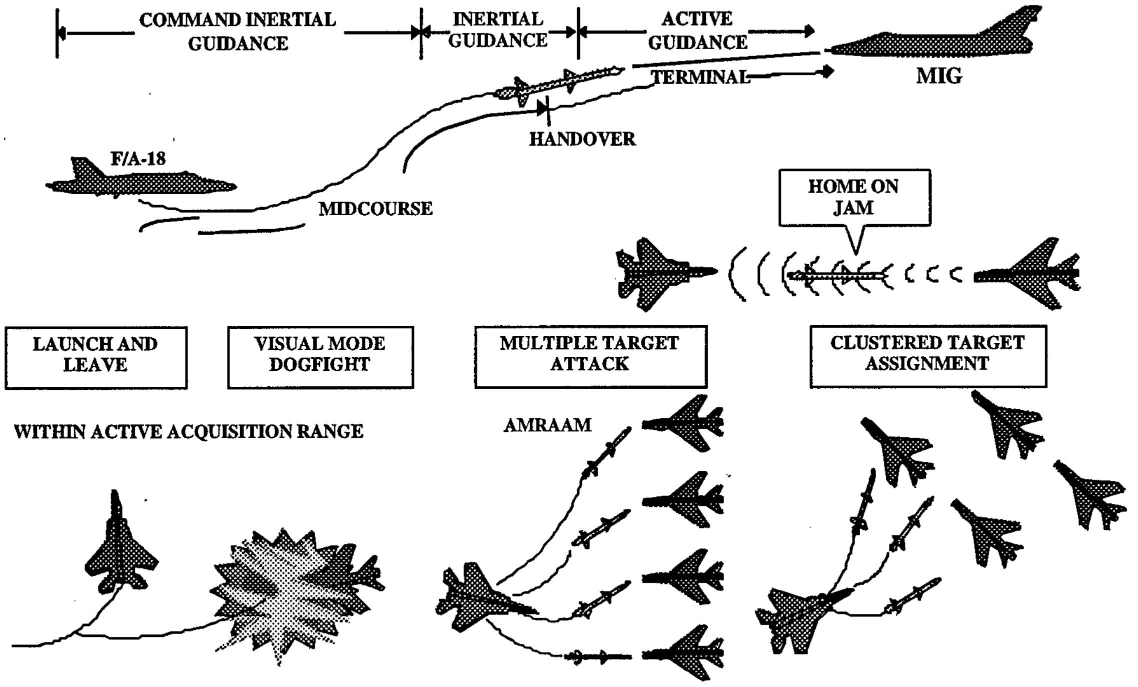


Figure 3
AMRAAM Launch Modes

The design of AMRAAM makes it very dependent on the quality of targeting information provided by the host aircraft. The requirement for the missile to be very selective, in other words to hit only the desired target and not one in close proximity, forces the missile to follow instructions from the host aircraft very carefully. Targeting information is provided by way of a one-way Radio Frequency (RF) data link that provides the missile with target information. Since the missile very closely follows host radar instructions, bad instructions will significantly reduce the effectiveness of the missile. Electronic jamming can cause significant degradation to the tracking ability of the host radar and thus degrade the quality of the targeting data passed via the data link. The AMRAAM searches a volume, known as the uncertainty volume, around the data link provided point. If the targeting data is bad enough the target will never be seen by the missile, even if it's own robust Electronic Protection capabilities would allow it to track and guide on the jamming target. This design philosophy is the key to the necessity of the test series.

3.2 Hardware in the Loop:

The AMRAAM Hardware-In-the-Loop (HWIL) lab located at the Naval Air Warfare Center in Point Mugu, CA provides the capability to simulate real-time missile flight scenarios from launch to target intercept. Housed in a 40' wide x 35' high spherical backplane anechoic chamber, the lab components include actual AMRAAM missile hardware, a CARCO flight motion table, computer-controlled Radio Frequency (RF) sources utilized to simulate maneuvering targets and electronic countermeasures, and computer systems that provide the real-time Six Degree-of-Freedom (6DOF) simulation. A brief description of each follows:

- The AMRAAM missile hardware consists of the RF sensor system, guidance section and the telemetry section.
- The RF seeker is mounted on the CARCO 3-axis flight table, which simulates missile flight dynamics by providing role, pitch and yaw motion. The CARCO table is positioned on a vibration isolated concrete pillar to prevent table dynamics from affecting the RF array.
- The RF source is a horn array of dual-polarized RF horns (vertical and horizontal polarization) that are positioned on the wall opposite the missile seeker. Through radiation emissions, the four target signal generators and three target positioning systems (TPS) provide the capability to simulate multiple targets, electronic countermeasures (ECM), Mainlobe Clutter (MLC) and Altitude Return (AR). The array of horns are positioned exactly 31.5 feet from the missile seeker to present a far field planar wave to the seeker. Figure 4 shows the RF horn array and flight table.

A. The RF sources can be arranged in the following ways:

1. One or two complex targets (including amplitude scintillation and angle glint) with or without ECM with or without Jet Engine Modulation (JEM), MLC and AR.
2. One or two complex targets with or without ECM with or without JEM with a Stand off Jammer (SOJ) and either MLC or AR.

B. The system is expandable to four more target signal generators (TSG) and five more target positioning systems for a total of eight TSGs and TPSs. This would allow six complex targets with ECM and JEM or four complex targets with ECM, JEM, MLC, and AR.

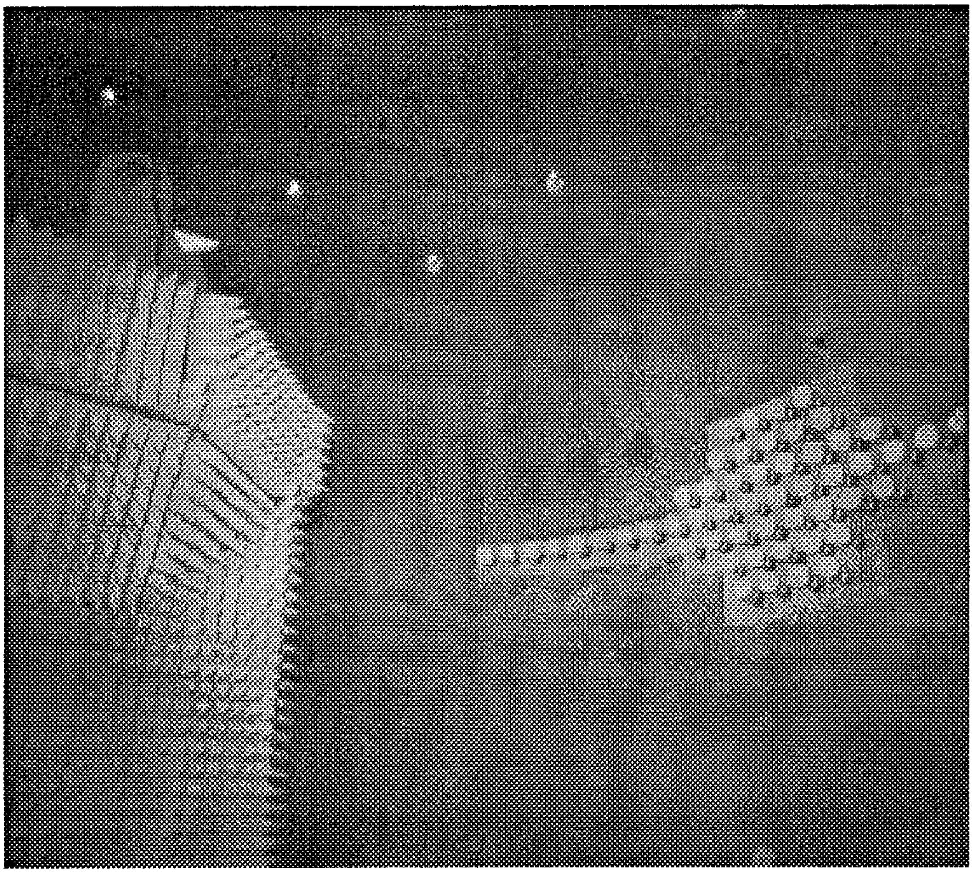


Figure 4
HWIL RF Horn Array

- Real-time simulation and modeling are provided by eight Intel i860 simulation computers operating in parallel, 2 SUN SPARC host computers, and a separate VME-based i860 data capture computer. In addition, the simulation system includes a Silicon Graphics (SGI) 3D computer used to display flyout graphics and an IBM compatible computer in conjunction with a Digital PDP-11 which control the simulated launch of AMRAAM. Collectively, all 14 computers are known as the Simulation Computer System (SCS). The 6DOF running on the SCS receives data from the AMRAAM missile that is stimulated by RF emissions from the array, then outputs kinematics control signals to the 3-axis flight table. The SCS also simultaneously drives the RF sources to replicate dynamic targets and the changing countermeasure and clutter environment. Figure 5 shows a basic functional diagram of the HWIL facility:

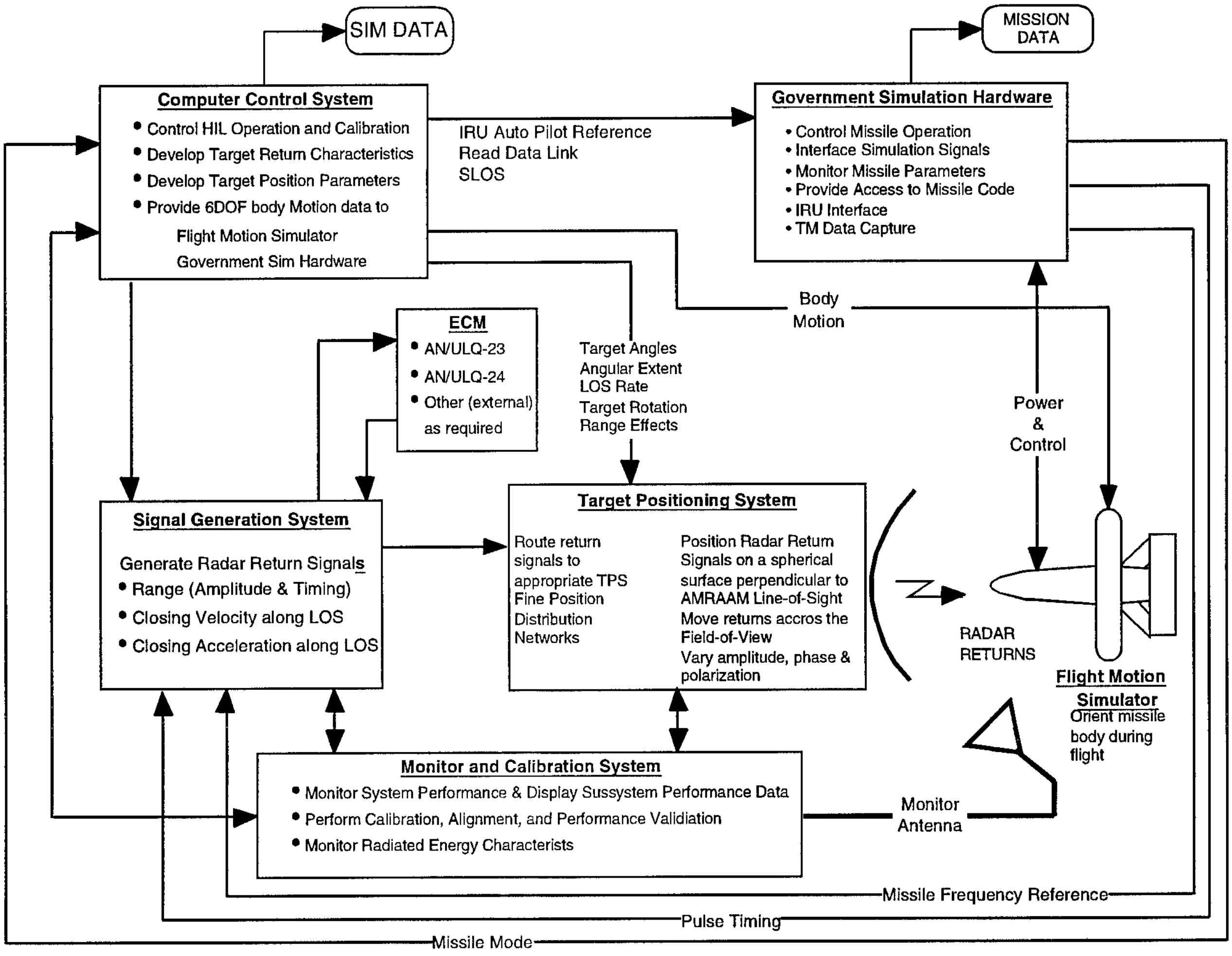

SIM DATA
MISSION DATA
Computer Control System
• Control HIL Operation and Calibration
• Develop Target Return Characteristics
• Develop Target Position Parameters
• Provide 6DOF body Motion data to Flight Motion Simulator Government Sim Hardware
IRU Auto Pilot Reference
Read Data Link
SLOS
Government Simulation Hardware
• Control Missile Operation
• Interface Simulation Signals
• Monitor Missile Parameters
• Provide Access to Missile Code
• IRU Interface
• TM Data Capture
Body Motion
ECM
• AN/ULQ-23
• AN/ULQ-24
• Other (external) as required
Target Angles
Angular Extent
LOS Rate
Target Rotation
Range Effects
Power & Control
Signal Generation System
Generate Radar Return Signals
• Range (Amplitude & Timing)
• Closing Velocity along LOS
• Closing Acceleration along LOS
Target Positioning System
Route return signals to appropriate TPS Fine Position Distribution Networks
Position Radar Return Signals on a spherical surface perpendicular to AMRAAM Line-of-Sight
Move returns accros the Field-of-View
Vary amplitude, phase & polarization
RADAR RETURNS
Flight Motion Simulator
Orient missile body during flight
Monitor and Calibration System
• Monitor System Performance & Display Sussystem Performance Data
• Perform Calibration, Alignment, and Performance Validiation
• Monitor Radiated Energy Characterists
Monitor Antenna
Missile Frequency Reference
Pulse Timing
Missile Mode

Figure 5
HWIL Facility Diagram

The primary purpose of the HWIL is to evaluate the performance of the AMRAAM missile seeker and guidance subsystem. The lab is utilized to perform both pre-flight and post-flight simulations for all planned F/A-18 flight test scenarios as well as AMRAAM specific testing and system level weapons system performance testing. A critical validation of the laboratory was performed to characterize technical performance parameters and identify critical limitations of the HWIL in order to identify simulation artifacts in test results.

The AMRAAM HWIL has recently gone through a major upgrade program. The aforementioned RF horn array, target signal generators, target positioning system, and the simulation computer system were all part of the upgrade. In addition, the upgrade included a new mainlobe clutter model, JEM model, and very robust calibration and diagnostic software. This upgrade provides the AMRAAM HWIL with the capability to be a valuable asset for AMRAAM activities for the foreseeable future.

3.3 F/A-18 WSSF:

The Navy is performing overall improvements of the F/A-18 weapon systems to keep pace with the evolutionary changes in digital technology, technical obsolescence, and operational requirements. The F/A-18 Weapon System Support Activity (WSSA) designs, develops, maintains, and operates integration and test facilities used in the development, test, evaluation, and Fleet support of the F/A-18 weapon system.

The F/A-18 WSSA Facility is the primary tool used to perform life-cycle maintenance of the F/A-18 weapon system embedded computer systems and associated Operational Flight Programs. It is comprised of a group of hardware-in-the-loop integration and test laboratories used to perform development testing, safety-of-flight testing, verification and validation testing, quick response investigations of Fleet reported problems, correction of deficiencies, investigation and evaluation of changes, and integration and test of new technology and weapons. These laboratories are continually evolving to support past, present, future, and foreign versions of the F/A-18 weapon system. Figure 6 shows the basic architecture of the WSSF facility.

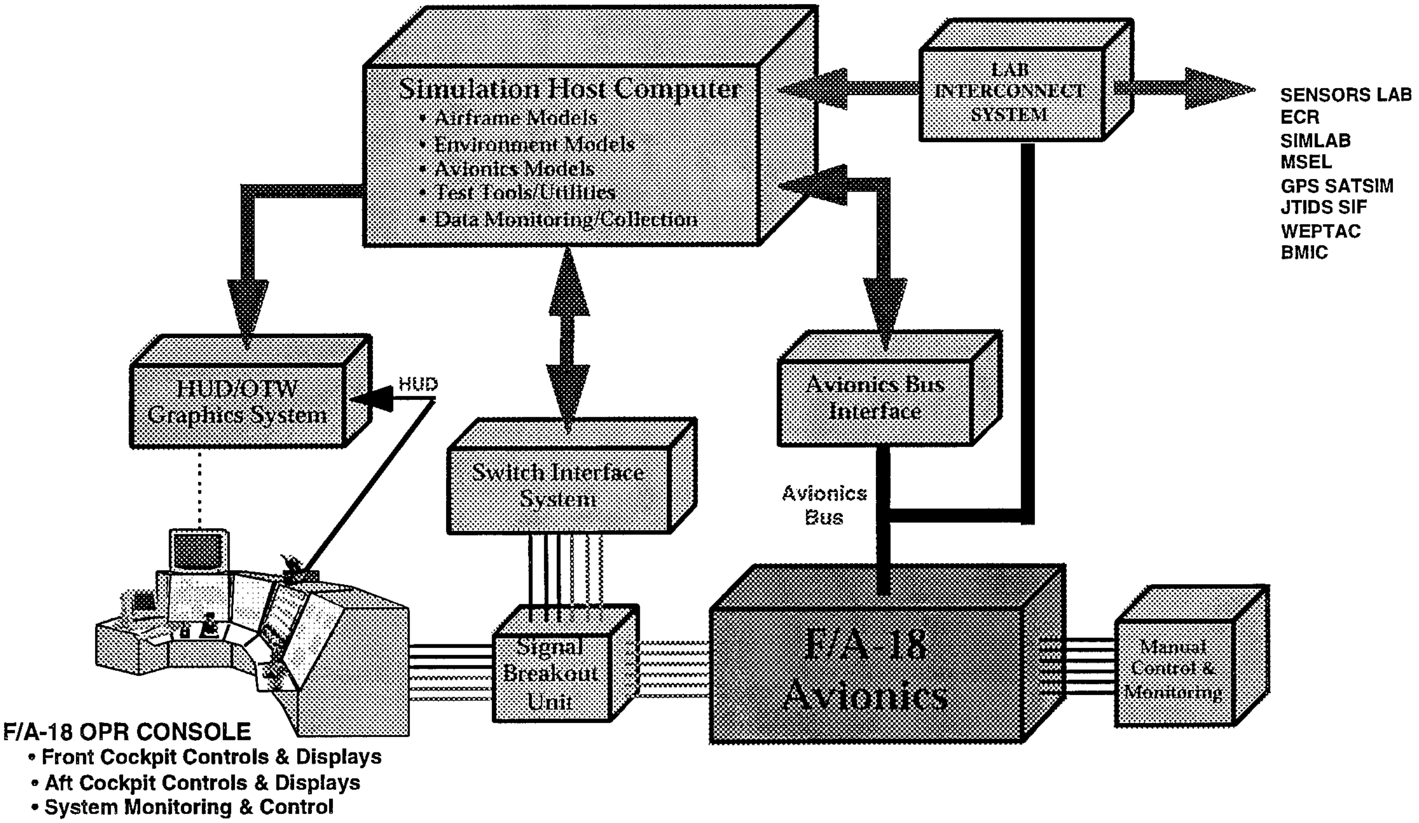


Figure 6
F/A-18 WSSF Architecture

Each laboratory uses real-time simulation to stimulate actual weapon system elements into "thinking" they are flying in a fully operational environment. Weapon system elements not desired in the laboratory, considered ineffective in a laboratory environment or unavailable are simulated using real-time mathematical models to affect a complete system during testing. On-line and post-test data monitoring and analysis tools provide in-depth visibility to assess system operation and performance.

Core mission system avionics and sensor laboratories (Radar, FLIR, EW, etc.) are combined to satisfy a full range of test requirements. The primary key features of the WSSA facilities are:

Flight Modes - Provides man-in-the-loop, profile generator, canned scenario, and precise manual control modes of dynamic flight. Actual cockpit controls and displays used with dynamic out-the-window scene generation provide realistic real-time test scenarios.

Tactical Environment - Flight system simulates platform dynamics, tactical environment, and user-selectable weapons system elements to support scenarios with combined real and/or simulated avionics.

Weapons - The use of real weapons for ground checks and simulated weapons for dynamic captive carry, launch, and post-launch engagements.

Control, Monitoring, & Data Collection - Dynamic scenario setup and control, on-line data/event monitoring, and real-time data collection for post-test analysis. Real-time non-intrusive logic and data sampling.

Flight Test Data Playback - Playback of instrumented aircraft flight test data in the laboratory to recreate problems and validate corrections.

Dynamic Lab-to-Lab Links to Other Facilities & Ranges:

- Sensor Lab Links for Radar, FLIR, EW, and RECCE Sensor Integration
- Hardware-in-the-Loop Missile Lab Links for Missile Integration
- Electronic Combat Range Link for Live-Threat EW Testing
- GPS Satellite Simulator Link for GPS Receiver Integration
- Range Control Center Link for Real-time TSPI & Future TM Data
- NRaD System Integration Facility Link for MIDS Link-16 Integration & Test

Each laboratory is comprised of a digital simulation system with embedded processors for associated weapon system data path interfaces, operator console (engineering cockpit), computer visualization, performance and monitoring system, and associated fighter aircraft simulation software. All aspects of the F/A-18 master modes are supported including navigation, air-to-air weapon delivery, and air-to-ground weapon delivery. Simulation functions include F/A-18 aircraft dynamics, environment (earth, atmosphere, and targets); model substitution for selected weapon system elements, dynamic out-the-window computer graphics, and specialized test functions. Figure 7 shows the relationship of the various F/A-18 development labs.

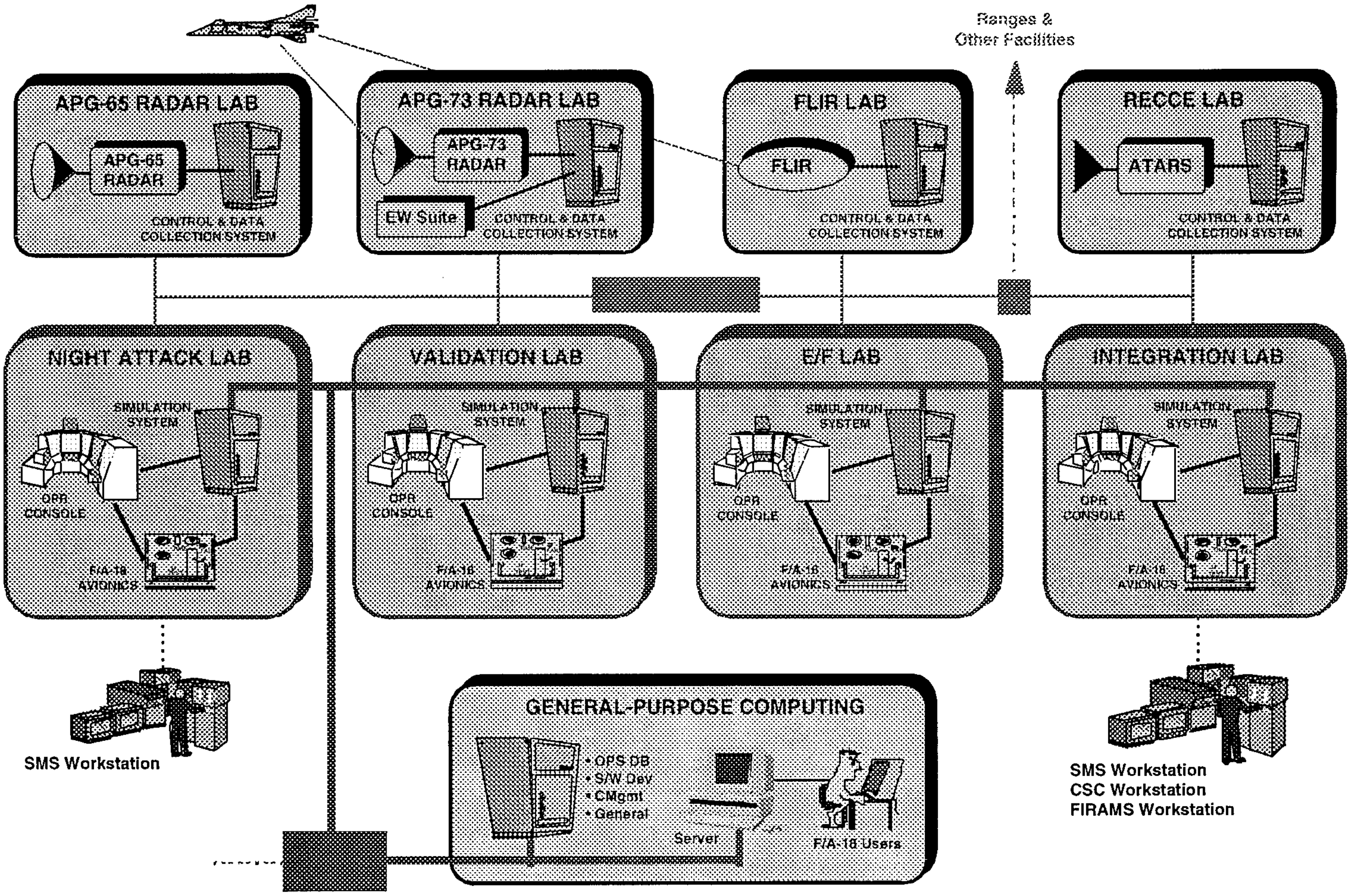


Figure 7
F/A-18 WSSF Lab Relationships

IV. Discussion

4.1 EPTWG:

In direct response to fleet requests for detailed weapons system performance information for the FA-18/AMRAAM combination, the F/A-18 and AMRAAM program offices funded a series of flight and simulation tests to gather information to meet fleet needs. Threats were chosen by the joint Navy-Air Force Electronic Protection Threat Working Group (EPTWG) which utilized inputs from the National Air

Intelligence Center and the EW Threat Simulations group at the Naval Air Warfare Center in Point Mugu, CA. The EPTWG consists of operational fighter aircrew from the Air Force, Navy and Marine Corps and is sponsored by the AMRAAM Joint Systems Program Office at Eglin AFB, FL. Threats were prioritized by evaluating threat effectiveness, proliferation and the likelihood of encountering them in combat. The EPTWG test series is currently in its third year and consists of flight testing of the aircraft radar against selected threats, AMRAAM Captive Equipment (ACE) flights against the same threats, Hardware-in-the-Loop simulation and digital simulation.

4.2 EPTWG Test Description:

The fidelity of test data is directly proportional to cost. The highest cost operation is, of course, a live launch but the data obtained is generally very enlightening. Captive flight testing is the next best thing and typically costs only about 25% of a live op. HWIL testing is the next highest fidelity test method. HWIL tests can be used to parametrically examine the affect of complex scenarios and environments. The EPTWG test series seeks to optimize available resources to obtain the highest fidelity data versus the most important threats and obtain the greatest volume of test data possible within budget constraints. Figure 8 depicts the cost relationships.

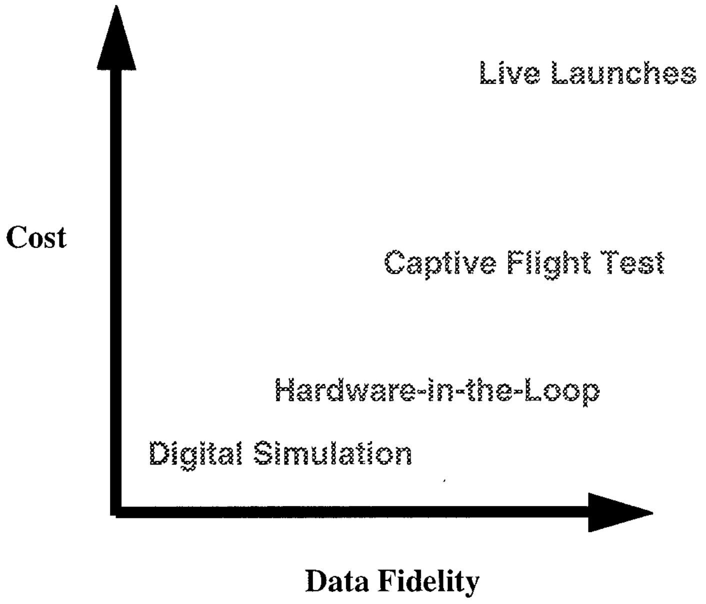


Figure 8
Cost versus Data Fidelity

The EPTWG test series includes a combination of flight test, Hardware in the Loop tests and digital simulation. Flight testing is performed in two phases: the first is funded by the F/A-18 and consists of radar evaluation flights against prioritized threats assigned by the EPTWG. The aircraft are heavily instrumented and extremely detailed radar performance data versus various jamming waveforms is obtained. The aircraft either carry AMRAAM Integration Test Vehicles (ITV) or Missile Message Units (MMU) to capture AMRAAM data link messages during simulated launches. Time, Space and Position Indications (TSPI) are compared with aircraft supplied data link to determine the accuracy of the data link messages. The captured in-flight data link is placed in a database and utilized for both HWIL and digital simulations.

The second phase of flight testing is funded by the AMRAAM program and consists of flights against the highest priority threats. Flights are performed with the AMRAAM Captive Equipment (ACE) pod that contains actual missile components and environmental conditioning to allow for multiple launches. The aircraft treats the ACE pod as an actual live missile and provides all support including postlaunch data link. After simulated launch, the ACE pod receives the data link and, when in range of the target will activate its onboard transmitter and attempt to acquire the target. The ACE pod utilizes the same tactical software as an actual missile. All missile functions are captured and transmitted real time through secure telemetry channels. Post-flight analysis includes a detailed analysis of aircraft data link, missile functioning and trackfile activity and TSPI comparison of actual versus system targeting information. ACE flight data is utilized post-flight to both help verify simulation validity and to improve fidelity of the data link database.

A majority of the EPTWG test series is performed on a Sun or PC workstation utilizing a 6 degree of freedom digital simulation that emulates aerodynamic performance and missile seeker and guidance section performance. Dynetics developed the digital simulation program under contract to the AMRAAM program office. Data link files from representative flight tests are utilized in the simulation to improve fidelity. Post-flight simulations are run to replicate actual flight test parameters and the results are compared to help validate simulation accuracy.

The HWIL lab is utilized to test actual jamming sources versus the missile in a variety of scenarios. In the past, canned aircraft data link values were used to perform testing. Presently, if pre-recorded data link from actual flight test sorties is available, it is utilized and has improved the fidelity of test results significantly. Testing both duplicates those performed in flight test and expands the test matrix to include those jamming scenarios for which flight test funding was not available. The preceding section carefully detailed the functioning of the HWIL facility.

V. F/A-18 and AMRAAM Link

The ultimate goal of the link is to develop a system performance evaluation tool for all versions of AMRAAM and AMRAAM-capable F/A-18 aircraft. This linked system will be used to evaluate basic software logic and interface functionality, and to determine system capability in complex Electronic Countermeasure (ECM) environments. To accomplish this, the missile and aircraft/radar simulations must have a fully functional real-time interface, and must be presented with time-synchronized and functionally equivalent Radio Frequency (RF) environments. The missile and aircraft/radar simulations must be of sufficient fidelity for both F/A-18 and AMRAAM programs to accept the results as flight-representative. Once this capability has been achieved, the combined link can be used to spot check actual flight results, and to supplement or replace some flight test missions for programs such as the Operational Tactics Guide (OTG) and Electronic Protection Threat Working Group (EPTWG) evaluations.

Initial work on the idea was begun in 1996 and initially consisted of a simple kinematic link implemented over a 150 plus mile fiber optic link. The kinematic link placed the HWIL missile in the same orientation as the aircraft at launch. For example, if the aircraft were in a turn the missile body in the HWIL would roll to match orientation to the aircraft. Prior to each linked run, a mutually agreed upon test scenario was chosen. The WSSF initiated a HWIL missile launch at a predetermined range to the target. The HWIL recognized the launch command and then passed canned umbilical and data link information to the missile that were specific to the chosen scenario.

The current HWIL/WSSF link provides for real-time kinematic interaction between the missile and aircraft simulations. In addition, a HWIL missile launch can be initiated from the aircraft cockpit simulation at the WSSF. In the current configuration, umbilical traffic and RF data link information are passed over a separate ethernet link which emulates MIL-STD-1553 format. The ethernet link allows passage of bi-directional real-time umbilical traffic and one way passage of aircraft data link information while the missile is in simulated free flight.

5.1 HWIL/WSSF Link System Architecture:

The hardware at both sites is connected by a fiber optic ethernet link over which aircraft and missile dynamics are passed. Aircraft supplied data link targeting messages are also passed over the ethernet link. A separate RS-232 Integrated Services Digital Network (ISDN) connection is established over which 1553 umbilical prelaunch messages, aircraft release consent and the interlock return confirming missile readiness to launch and the actual missile launch. The ISDN link is controlled by PC's at each location emulating a 1553 bus. Both ISDN and ethernet connections are encrypted. Figure 9 shows the basic relationship of the labs and link implementation.

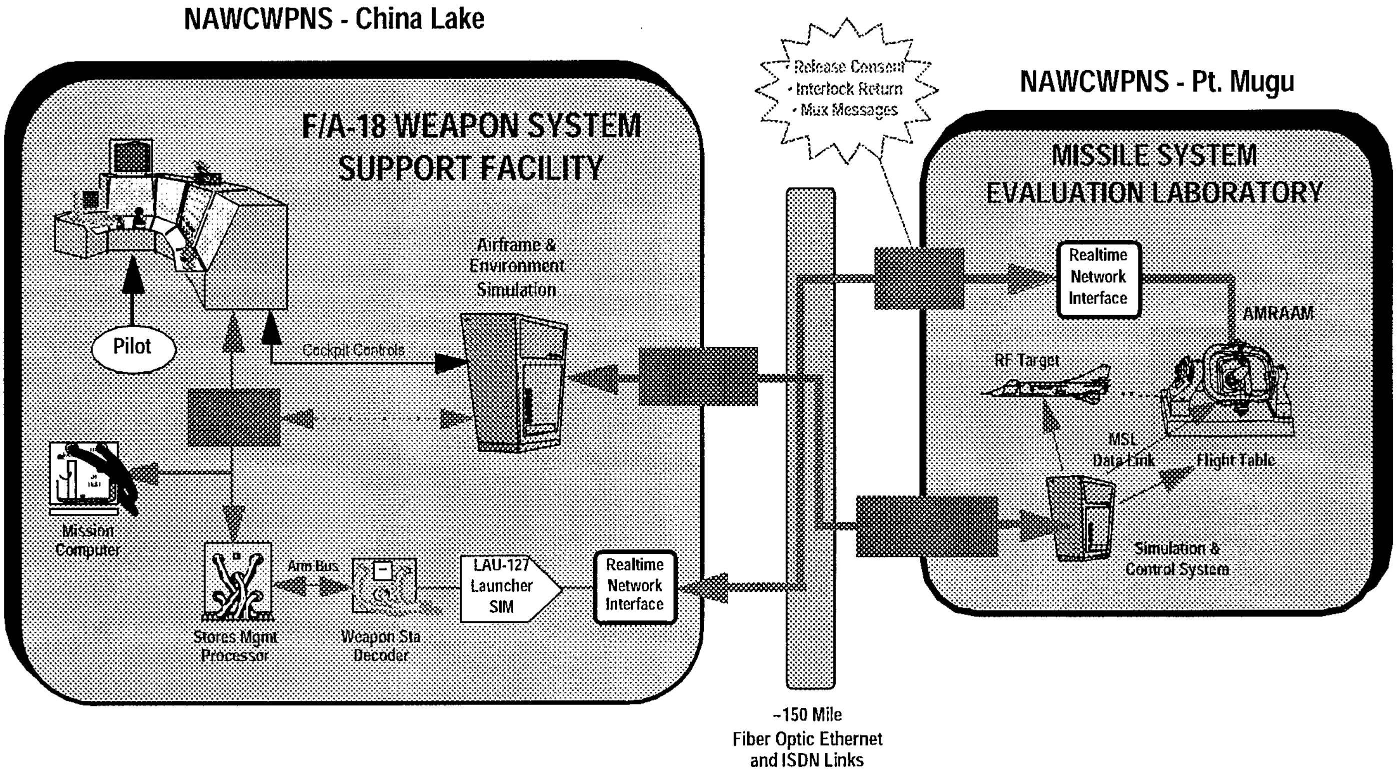


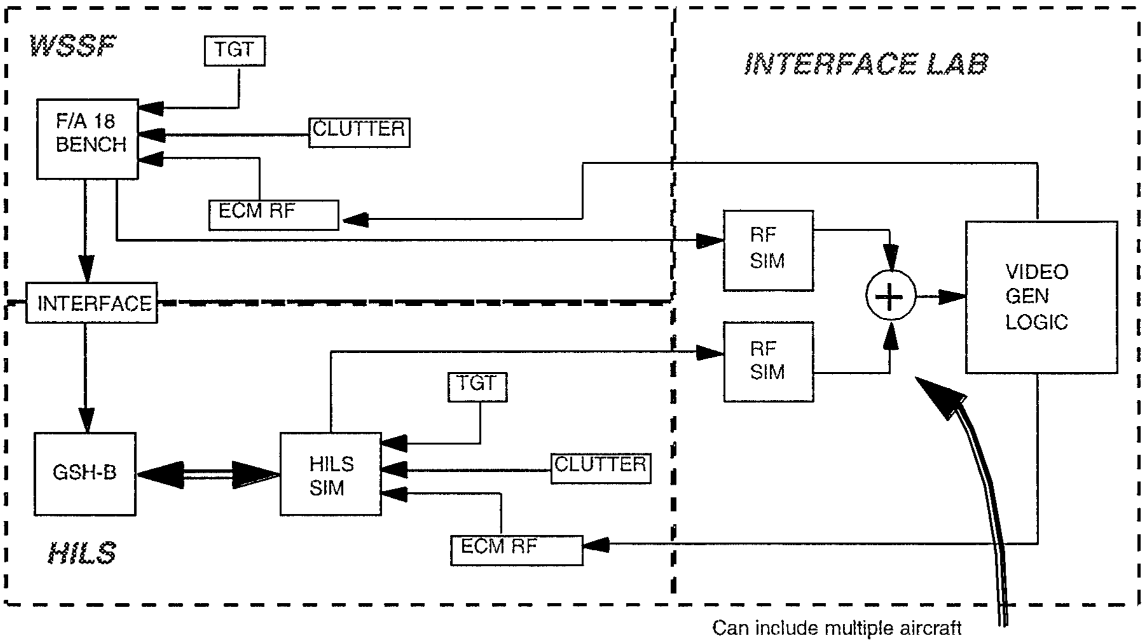
Figure 9
WSSF and HWIL Link Concept

5.2 Future Enhancements:

The system as currently implemented is quickly maturing towards utility for system level testing. Currently, efforts are underway to both improve the existing implementation and increase capabilities in two critical areas. The first is the capability to use the actual radar in the WSSF to present real time targets to the AMRAAM lab. The second critical area is the capability to present matched RF and jamming environments at both labs.

The capability to use the actual radar in the WSSF is the easier of the two remaining hurdles to solve. There are two ways this capability can be implemented. The first is to simply roll the radar out and track actual airborne targets operating on the local range. This capability exists now but requires actual airborne targets, which restricts testing time and greatly increases cost. The preferred method is to synthetically inject target signals into the WSSF either at the front end of the radar or downstream at Intermediate Frequency (IF). Both of these capabilities currently exist at the WSSF but have not had data blocks implemented for the ethernet link that is required to pass the target(s) to the AMRAAM HWIL. This represents a minor technical challenge and the capability should be available in the near term.

A significantly more complex issue is that of matching the RF environments at both sites. Several challenging technical problems must be resolved before the link will reach full functionality. The first problem is the time synchronization of the jammers at each location so that the missile and radar are presented the same environment with no data latency. The current plan is to implement a time synchronization scheme to time tag all traffic to the AMRAAM HWIL. In this implementation, data latency should not be an issue since everything will be referenced to the same time. Another possible solution is to not really worry about it since the delay should not exceed 100ms which is sufficient in most scenarios. One very difficult issue to resolve is providing a correct jammer response when the jammer is stimulated by both the F/A-18 and the AMRAAM simultaneously. Several ideas to solve this have been floated but none have as of yet, been tested. One solution is to only stimulate the jammer at the AMRAAM HWIL with multiple signals with the F/A-18 signals provided from a canned set of radar parameters that would be stored in the AMRAAM HWIL simulation and called up based on F/A-18 radar mode. Power levels would be simulated by modeling F/A-18 transmitter power out, antenna gain, radar mode and range to target. Figure 10 depicts a possible implementation.



Can include multiple aircraft
or missiles from ECM Simulation

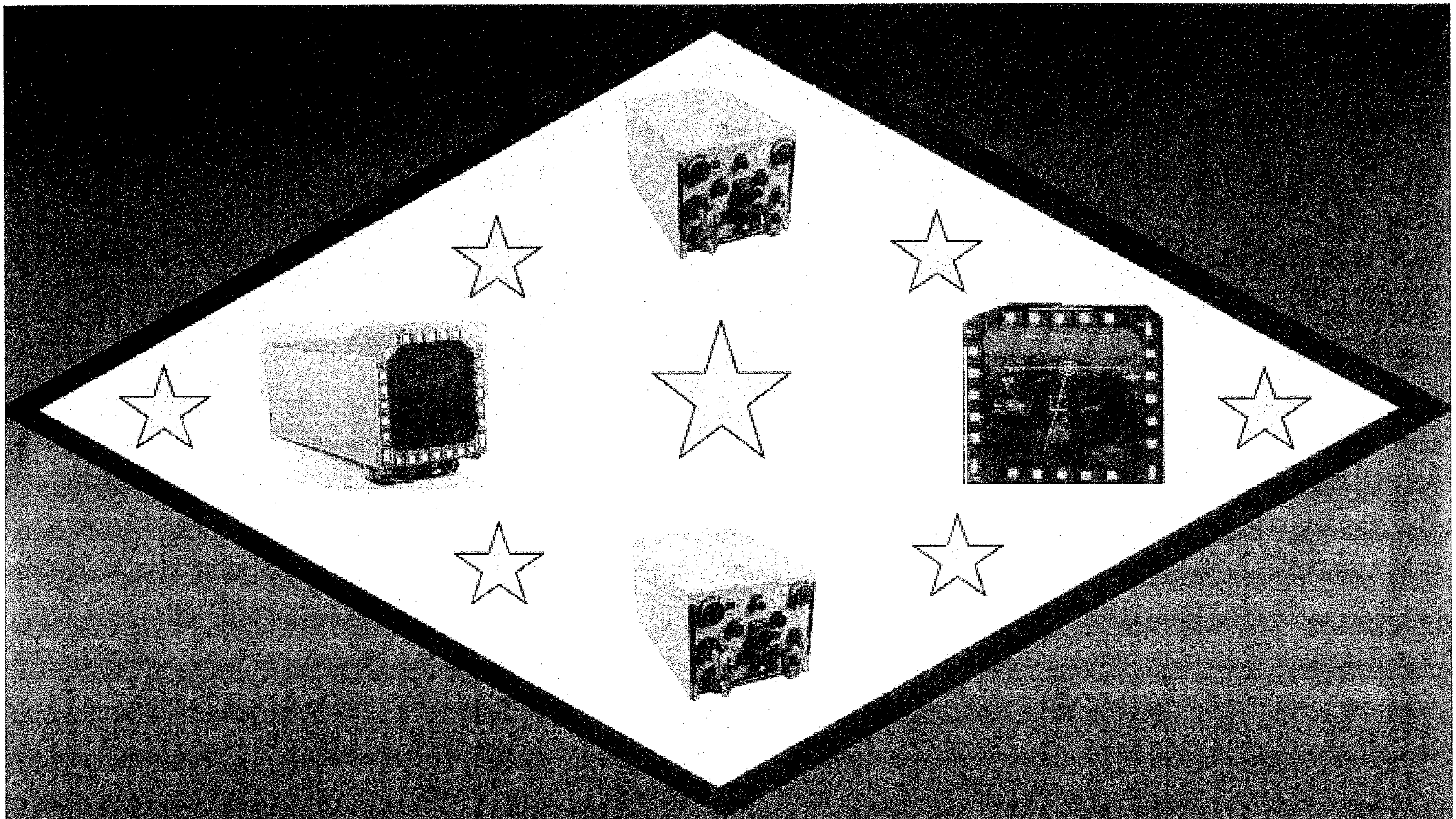
Figure 10
Proposed RF Synchronization Scheme

VI. Conclusions

The diminishing availability of funding for development and testing is forcing programs to reevaluate the manner with which they expend their limited resources. Recent advancements in computer processing power, digital simulation and telecommunications technologies have presented us with new opportunities to exploit to economize on testing and development.

Though the U.S. defense budget has declined significantly from the cold war era, the various threat systems around the world have continued to develop significant new capabilities, particularly in the arena of electronic warfare. Operational Fleet aircrews need the most up to date and accurate information available regarding the effectiveness and limitations of their weapons systems. Utilizing existing assets such as the AMRAAM HWIL facility and the F/A-18 WSSF more intelligently is critical to maximizing the weapons system data that can be collected within the available budget.

The AMRAAM HWIL and F/A-18 WSSF link takes a large step towards realizing the goal of performing economical system level testing to supplement and hopefully reduce the amount of flight testing required to satisfactorily address questions of system capabilities. The link represents a significant stride in the area of distributed simulation and an innovative use of existing simulation and development assets. The full future planned development of the link will make it not only a good tool for system level performance testing, but also for both F/A-18 Operational Flight Program development and AMRAAM enhancements and development.



ADVANCED MISSION COMPUTER & DISPLAYS

JAWS S^3 SYMPOSIUM & EXHIBITION

JUNE 15, 1998

Advanced Mission Computer & Displays

Kathy Talton	Ted Mitchell
PMA-209I	PMA-209P3

Address:

Commander
Naval Air Systems Command (Code: PMA-209I)
Naval Air Systems Command Headquarters
47123 Buse Road
Patuxent River, Maryland 20670-1547

Telephone:

Kathy Talton

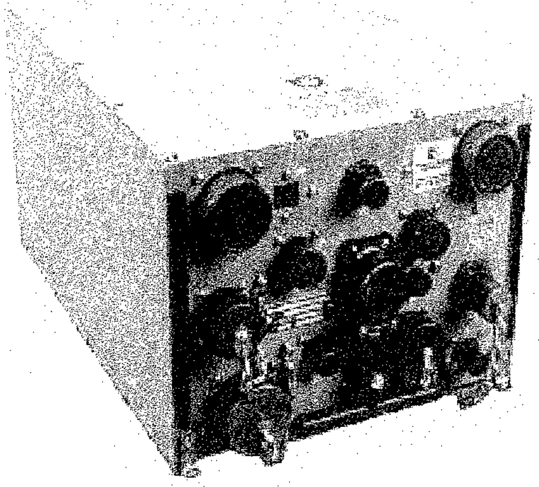
Commercial:	(301) 757-6771
Fax:	(301) 757-6459

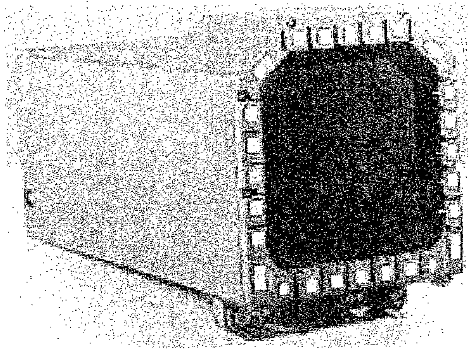
Ted Mitchell

Commercial:	(301) 757-6770
Fax:	(301) 757-6459

AMC&D OUTLINE

- Program Summary
- Objectives & Vision
- Requirements & Strategies
- Hardware
- Schedule
- Summary





AMC&D PROGRAM SUMMARY PMA-209

PROGRAM SUMMARY: Develop, procure, & support an Advanced Mission Processor, Display Processor and a family of Display Heads to replace current systems and support new requirements.

PLANNED USERS: F/A-18C/D/E/F (Lot 12 & above)

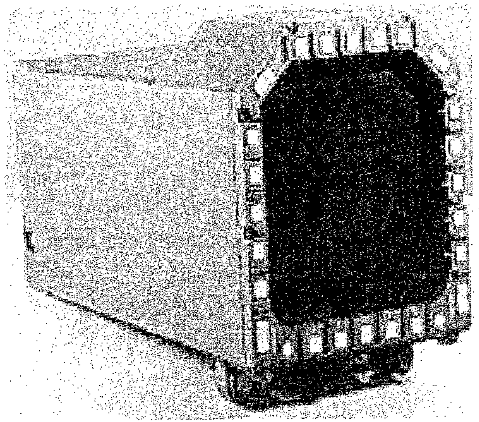
POTENTIAL USERS: V-22, AV-8B, F-15, SH-60R/CH-60, EA-6B, T-45, S-3, P-3, C-17, FMS, USA, USAF

QUANTITY: 1100 Planned Systems

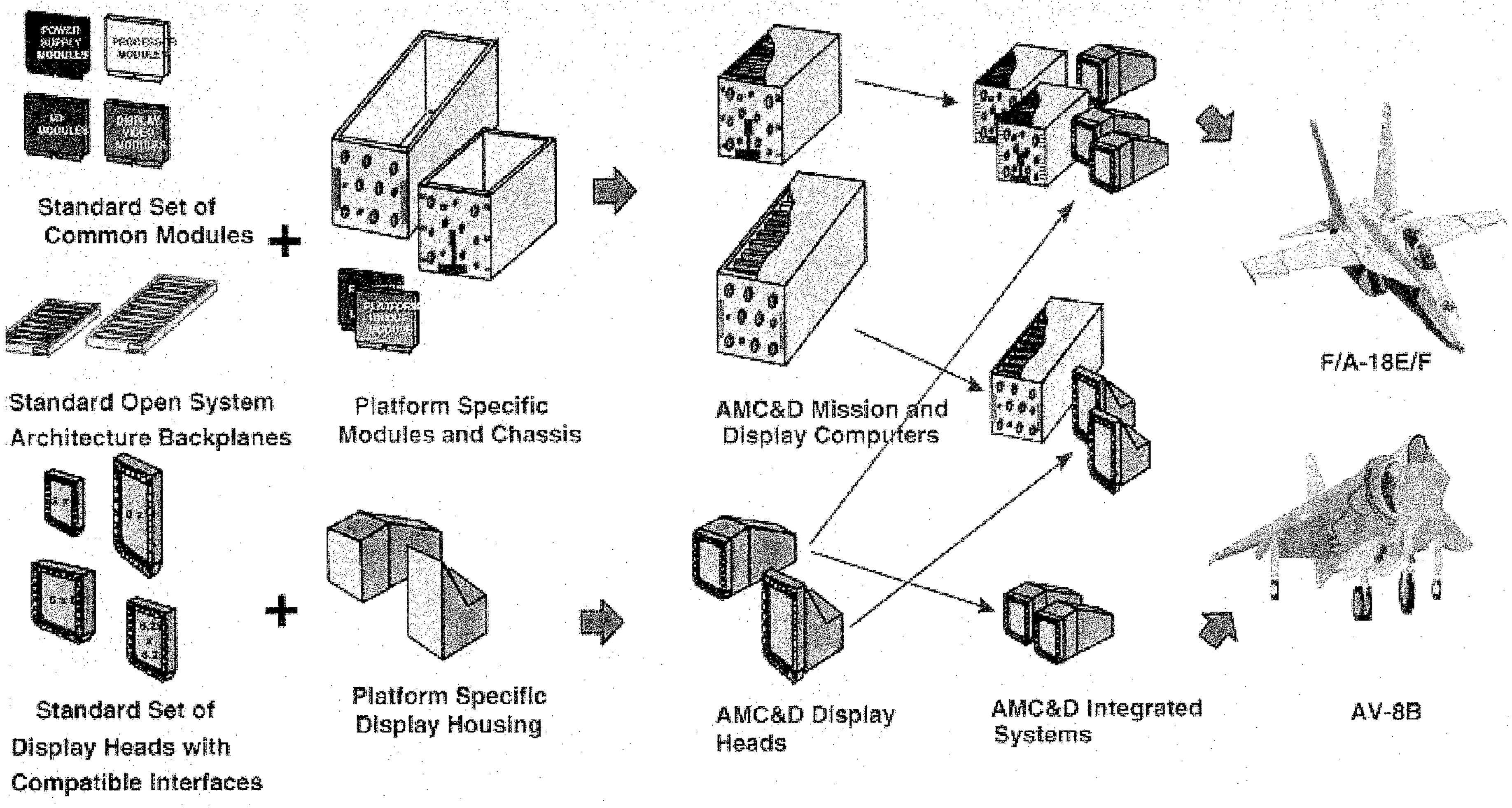
ACAT: ACAT II

PROGRAM OBJECTIVES

- Develop a family of common mission & display processing systems and family of display heads that meet the needs of naval aviation & other users
- Use an open systems architecture
- Minimize integration impact to the aircraft
- Leverage commercial technologies
- Obsolescence Management
- Decrease life cycle cost
- Design for growth



AMC&D — AMC&D VISION — PMA-209


POWER SUPPLY MODULES
PROCESSOR MODULES
I/O MODULES
DISPLAY VIDEO MODULES
Standard Set of Common Modules
+
Standard Open System Architecture Backplanes
PLATFORM UNIQUE MODULES
Platform Specific Modules and Chassis
AMC&D Mission and Display Computers
F/A-18E/F
Standard Set of Display Heads with Compatible Interfaces
+
Platform Specific Display Housing
AMC&D Display Heads
AMC&D Integrated Systems
AV-8B

Select Appropriate AMC&D Subsystems

Add platform specific module and chassis

Select appropriate AMC&D Subsystems

Integrate into fully functional AMC&D Systems

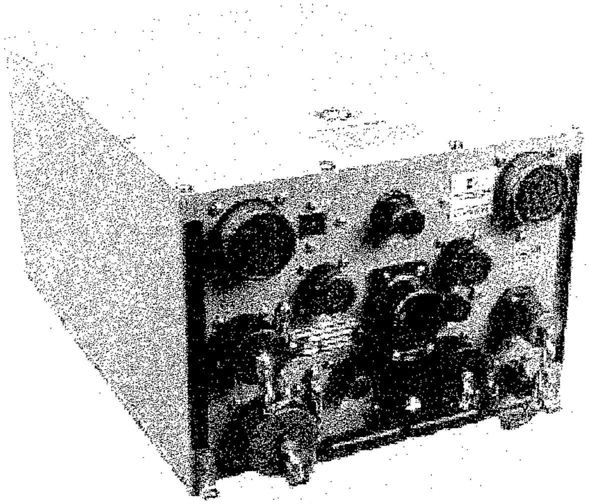
To meet Aircraft specific needs

AMC&D

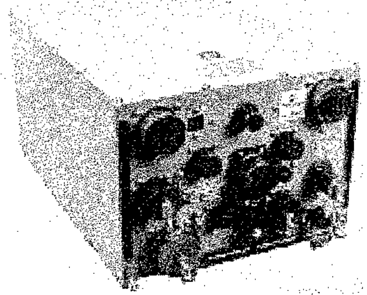
OPERATIONAL REQUREMENTS

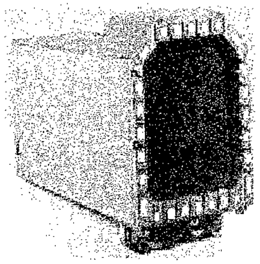
PMA-209

- Warfighting Requirements
  - » High resolution digital video imagery
  - » Improved processing
  - » Day & night enhancements
    - – Improved readability in sunlight
    - – NVIS compatible
  - » Color and monochrome modes
  - » Low power and weight
- Obsolescence
  - » CRT technology reaching end of life
  - » Processor and memory requirements double every 2-3years.
- Affordability
  - » Reduce O&S costs
  - » Improve system reliability
  - » Reduce system recurring costs



AMC&D TECHNICAL/LOGISTICS STRATEGY PMA-209





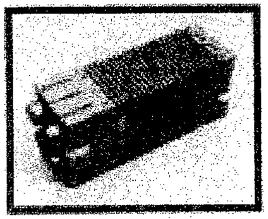
- Performance Specifications
- Open System Architecture
  - » VME64 Backplane (6U format)
  - » POSIX compliant Operating System
  - » HOL Support
  - » Support use of commercial software tools
  - » Non-proprietary interfaces
- Establish common requirements
- Draw from on-going programs and technology efforts
- Perform trade studies for technology decisions
- Modular design concepts
- Maximize use of COTS/NDI products
- Two level maintenance (O to OEM)
- Stringent BIT requirements
- Warranty

AMC&D

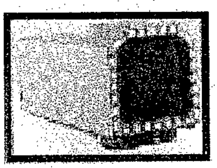
AMC&D HARDWARE

PMA-209

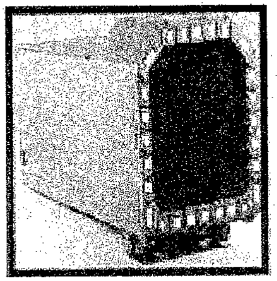
High Speed Data Network Switches (Fibre Channel)



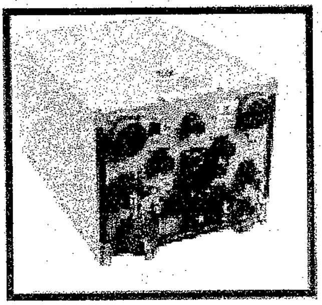
5x5 Display



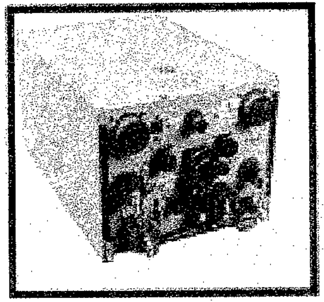
8x10 Display



Mission Processor (MP)



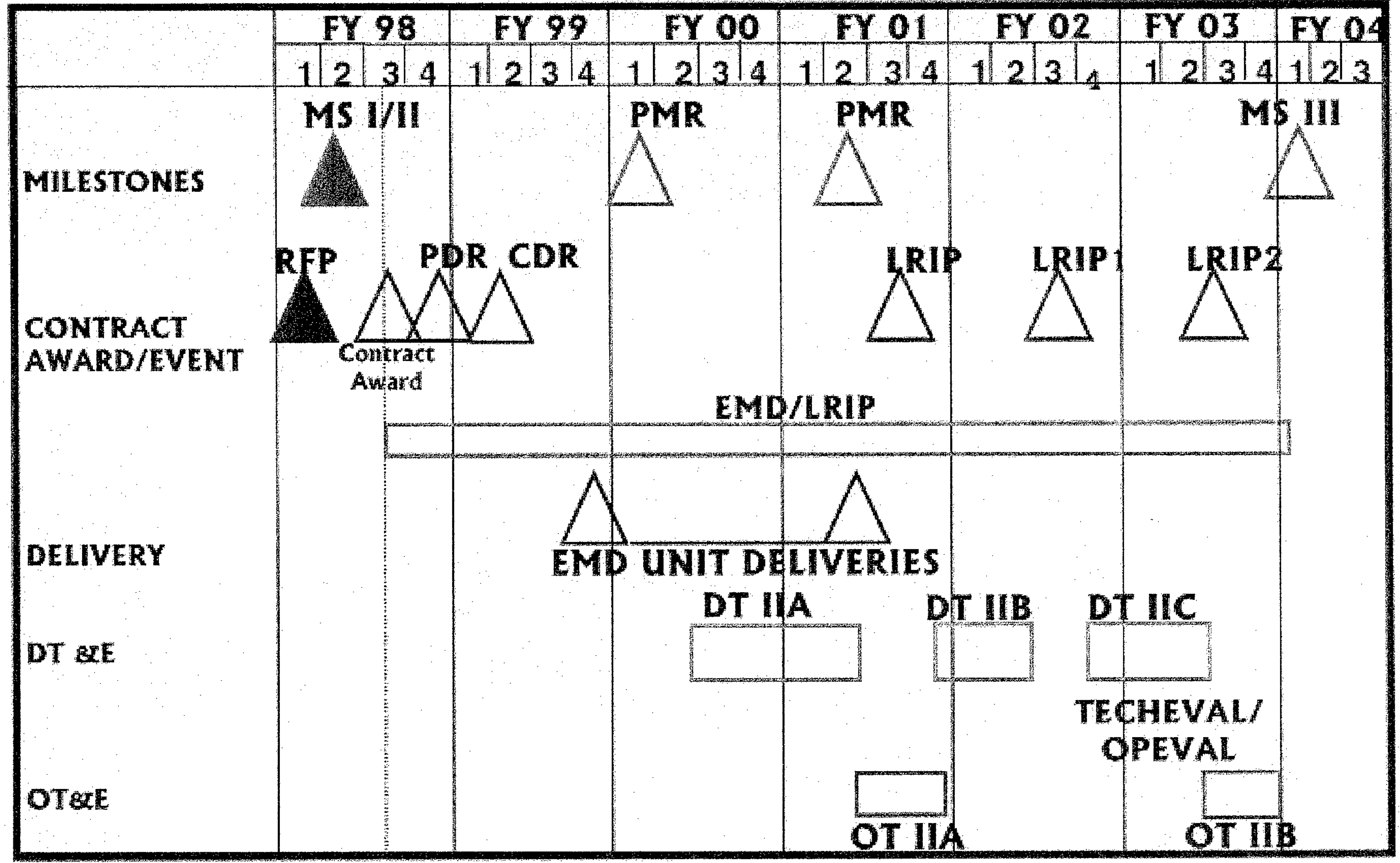
Display Processor (MP)



AMC&D

MASTER PROGRAM SCHEDULE

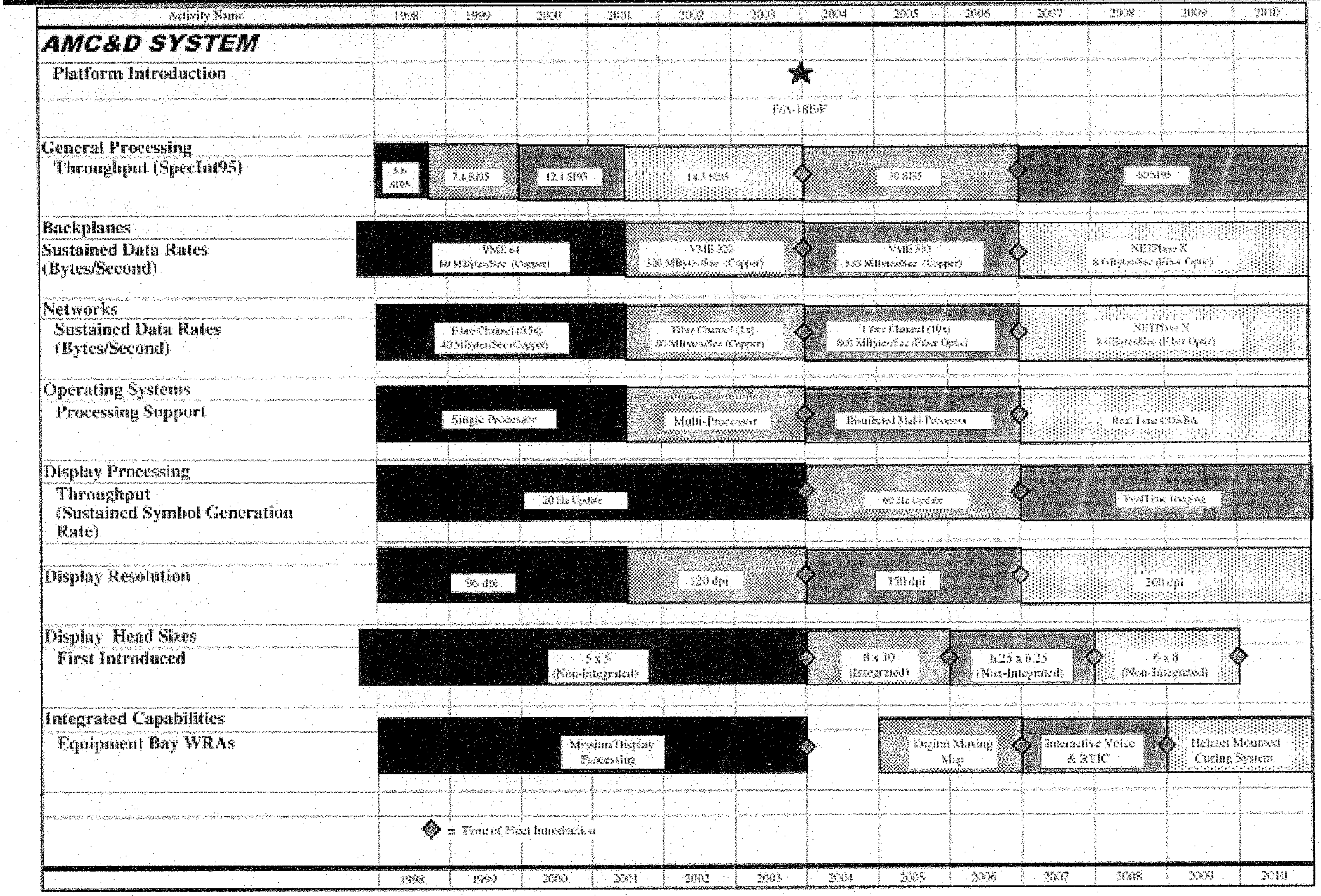
PMA-209



AMC&D

AMC&D ROADMAP

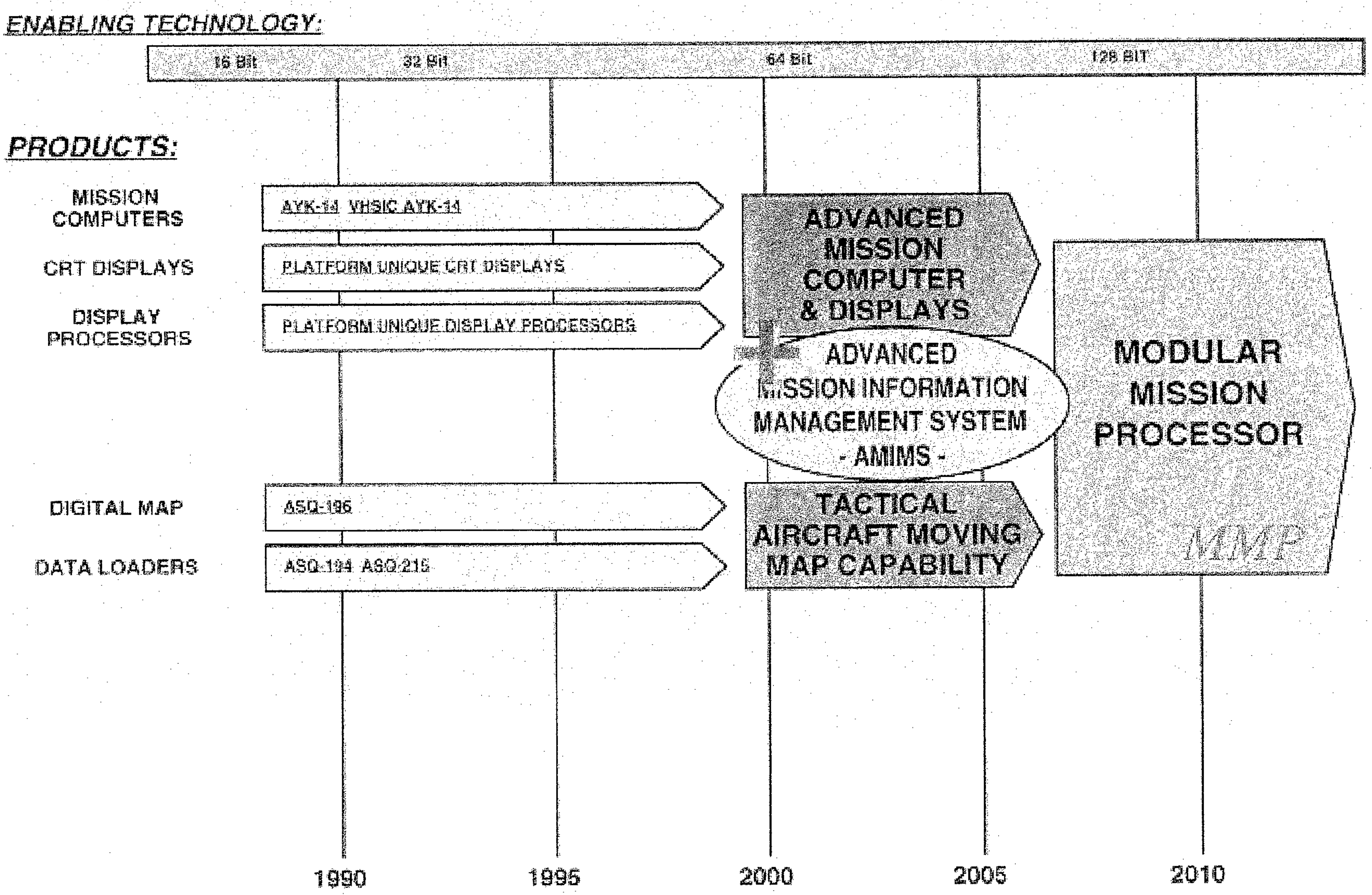
PMA-209



AMC&D

PMA-209 ROAD MAP

PMA-209



ENABLING TECHNOLOGY:

16 Bit

32 Bit

64 Bit

128 BIT

PRODUCTS:

MISSION COMPUTERS

AYK-14 VHSIC AYK-14

CRT DISPLAYS

PLATFORM UNIQUE CRT DISPLAYS

DISPLAY PROCESSORS

PLATFORM UNIQUE DISPLAY PROCESSORS

DIGITAL MAP

ASQ-196

DATA LOADERS

ASQ-194 ASQ-215

ADVANCED MISSION COMPUTER & DISPLAYS

ADVANCED MISSION INFORMATION MANAGEMENT SYSTEM - AMIMS -

TACTICAL AIRCRAFT MOVING MAP CAPABILITY

MODULAR MISSION PROCESSOR

MMP

1990

1995

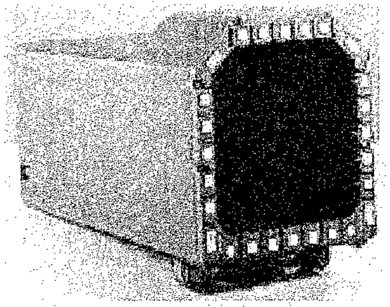
2000

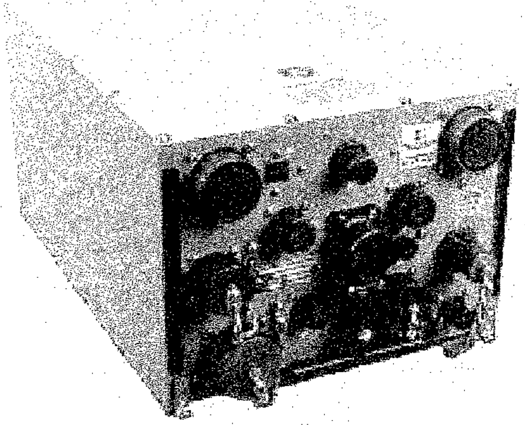
2005

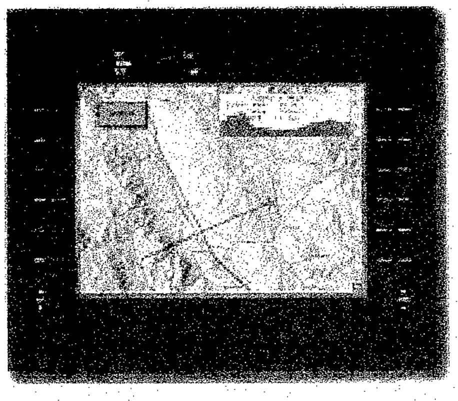
2010

SUMMARY

- Increased capability, improved reliability, and obsolescence solution
- Open systems architecture
- Potential joint opportunities







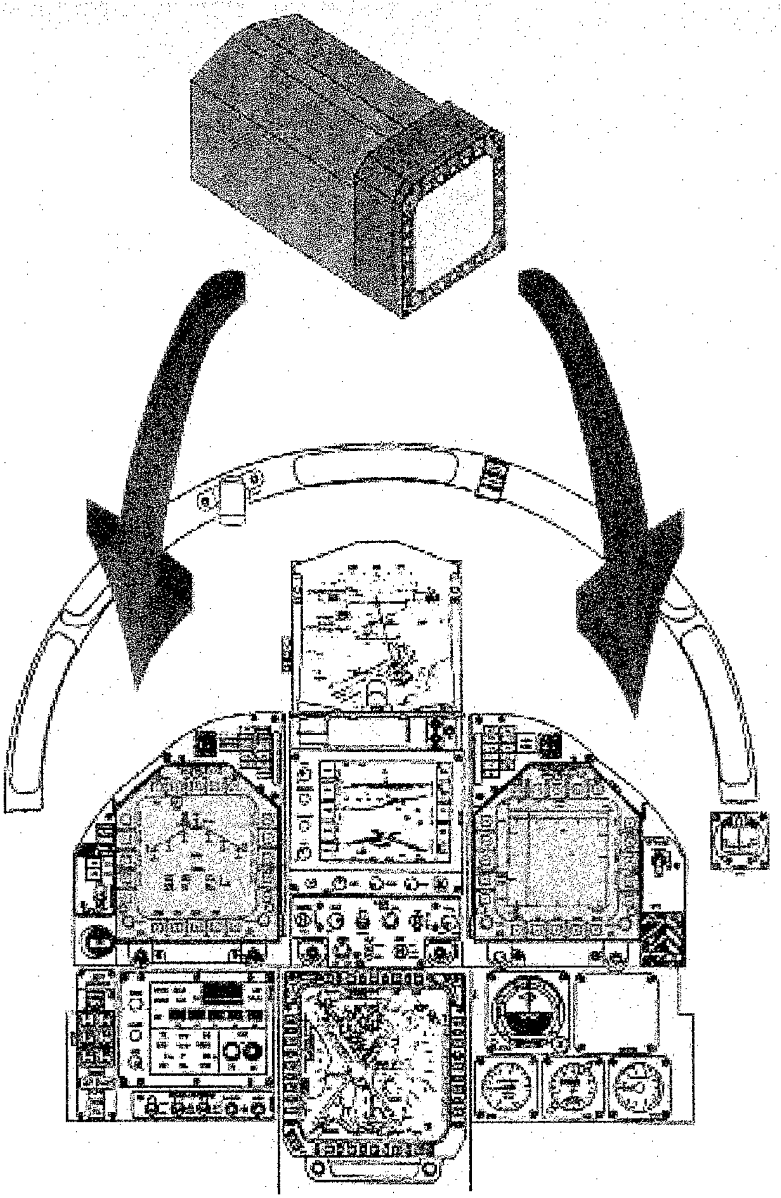
BACK-UP

AMC&D

5 x 5 AMPD

PMA-209

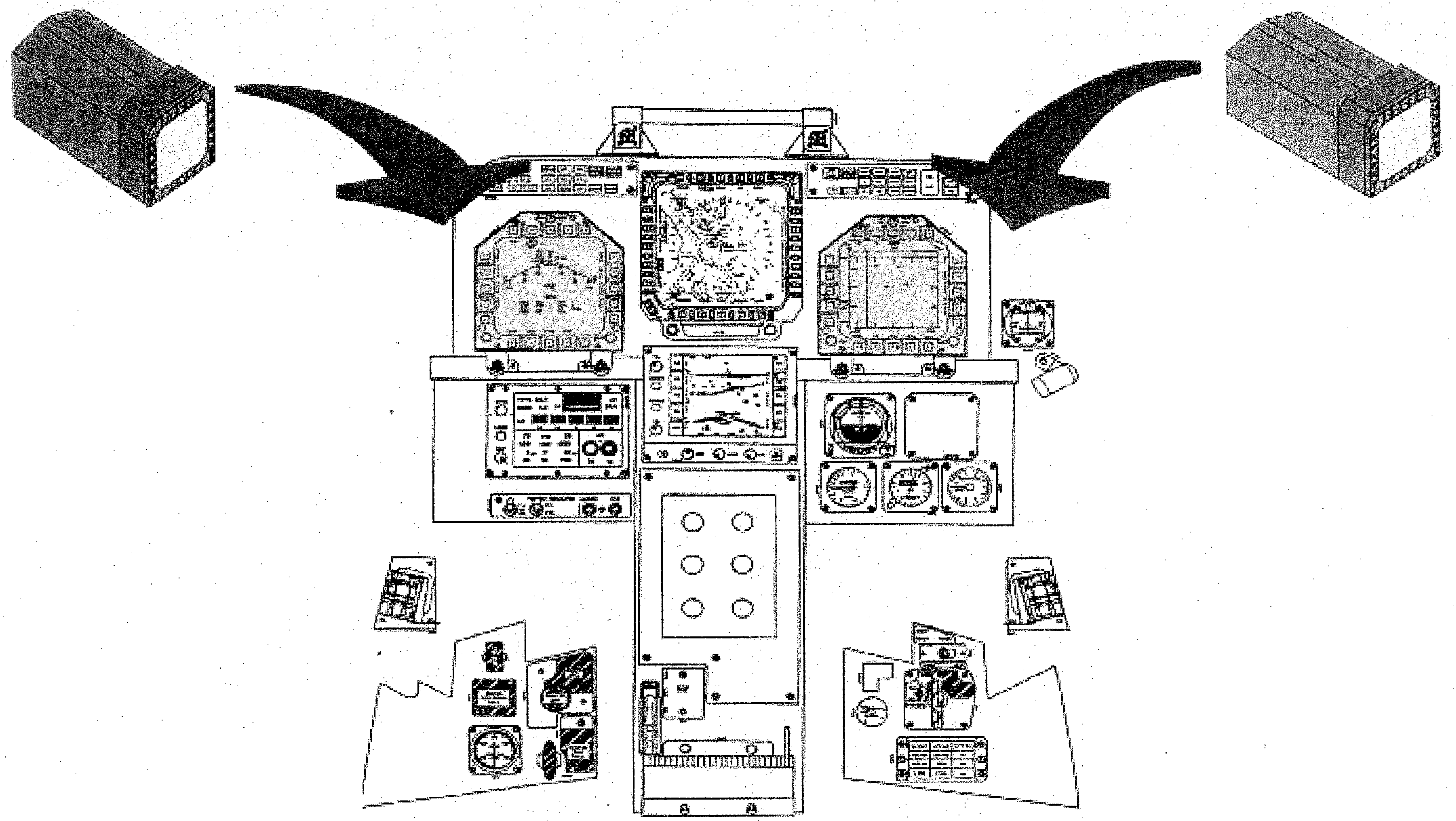
- High Resolution / Flat Panel LCD display
- Replaces legacy CRT displays
- Maintains basic outline of legacy displays
- No write-time limitations
- Common Display Elements
  - » **Fore/Aft Commonality**
- Full Color Displays
- High Contrast Ratio Displays
  - » **High Ambient Atmosphere**



AMC&D

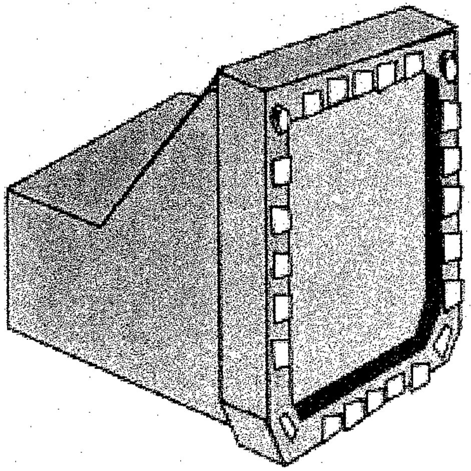
AFT INSTALLATION

PMA-209



8 x 10 AMPD

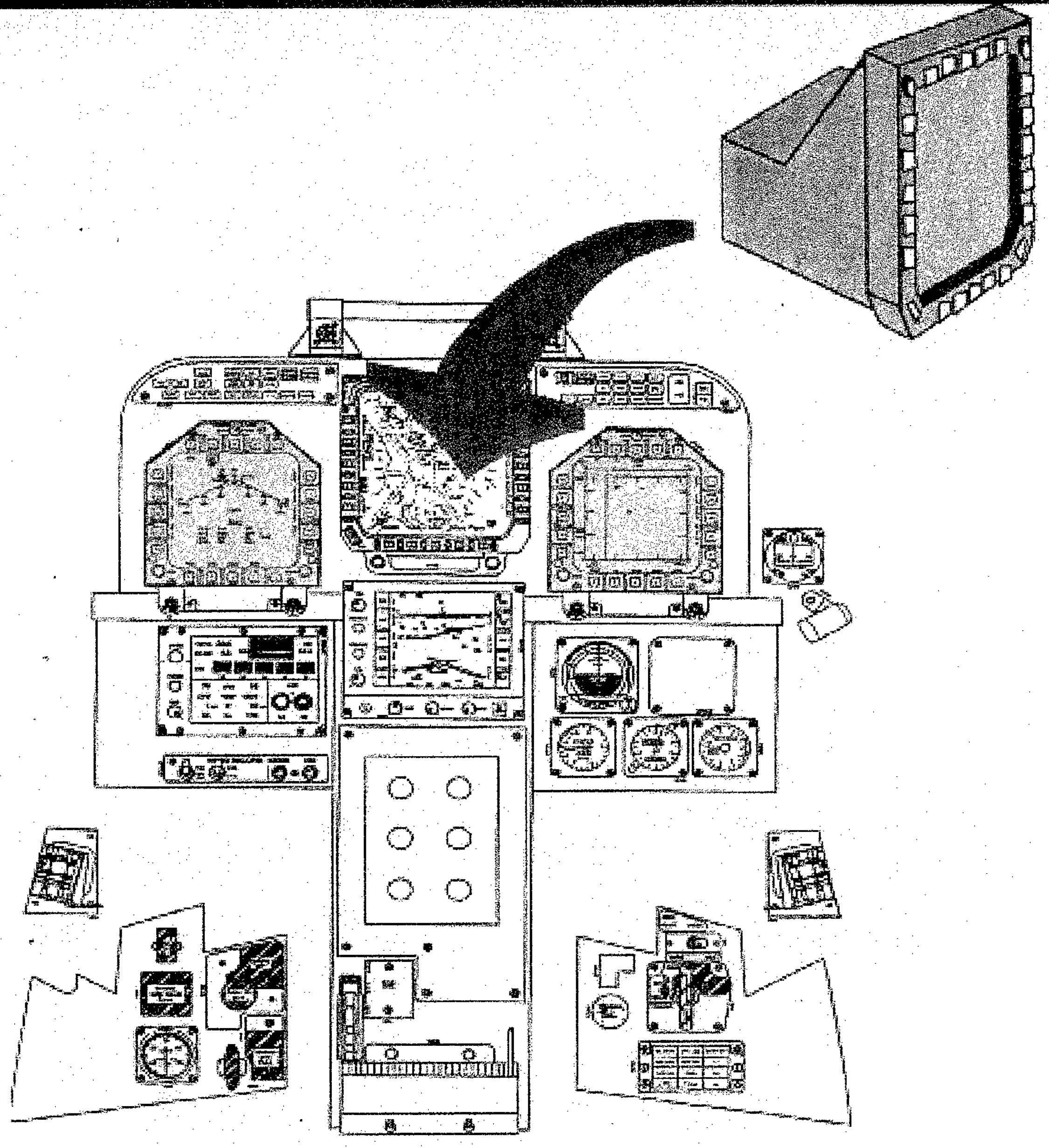
- 8"x10" Display Surface
- Full Color, High Contrast Ratio AMLCD Display
- Replaces Aft Multipurpose Color Display
- Improved S.A.
- Requires redesign of Aft Crewstation



AMC&D

8x10 AMPD INSTALLATION

PMA-209



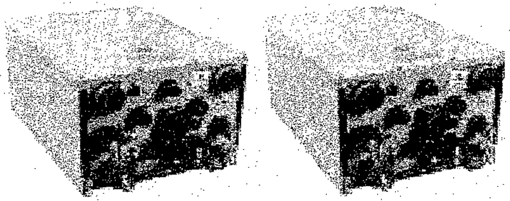
AMC&D

SYSTEM COMPARISON

PMA-20

Old

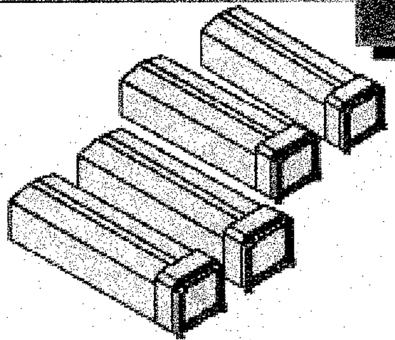
Mission Computers (AYK-14)



- 2 Boxes
- 8-10 MIPS
- 8 MB Memory
- 6 1553 Channels
- 10.3"x7.6"x14" ea.
- 76 lbs Total
- 560 Watts Total
- $ 322K* Total

Multipurpose Display Indicators (MDI)

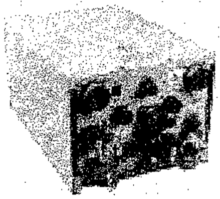
Multipurpose Display Repeater Indicators (MDRI)



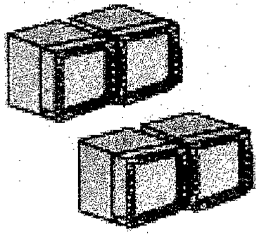
- 5"x5" Display Area
- 4 Video Inputs/ Outputs (MDIs)
- 3-Colors (Kroma LCD Filter)
- 119 lbs. Total
- 1,136 Watts Total
- $ 527K* Total

New

Mission Processor



Flat Panel Displays



Display Processor

- 2 Boxes
- 4 Display Heads (5X5)
- 480 MIPS
- 96 MB Memory
- 13 Video Inputs
- 8 Video Outputs
- 6 1553 Channels
- Full Color

- AYK-14 Form Factor
- 166 lbs. Total
- 1,141 Watts Total
- $ 750K* Total

Total Savings:
(Processors & Displays)

Cost:	$ 100K*
Weight:	29 lbs
Power:	555 W

*FY01 constant dollars